Technische Universität Ilmenau
Fakultät für Elektrotechnik und Informationstechnik

Dissertation

Self-Organizing Network Optimization via Placement of Additional Nodes

---

vorgelegt von:	Mikhail Tarasov
geboren am:	5.08.1986 in Moskau (Russland)
eingereicht am:	8. April 2013
Anfertigung im Fachgebiet:	Kommunikationsnetze
	Fakultät für Elektrotechnik und Informationstechnik
erster Gutachter:	Prof. Dr. rer. nat. Jochen Seitz
zweiter Gutachter:	Prof. Dr.-Ing. habil. Kai-Uwe Sattler
dritter Gutachter:	Prof. PhD. Timothy X. Brown

urn:nbn:de:gbv:ilm1-2014000017

Ilmenau University of Technology

Faculty of Electrical Engineering and Information Technology

Doctoral Thesis

Self-Organizing Network Optimization via Placement of Additional Nodes

---

Author:	Mikhail Tarasov
Born on:	5^{th} August 1986 in Moscow (Russia)
Submission date:	April 8, 2013
Department:	Communication Networks
	Faculty of Electrical Engineering and Information Technology
First adviser:	Prof. Dr. rer. nat. Jochen Seitz
Second adviser:	Prof. Dr.-Ing. habil. Kai-Uwe Sattler
Third adviser:	Prof. PhD. Timothy X. Brown

urn:nbn:de:gbv:ilm1-2014000017

Abstract

The main research area of the International Graduate School on Mobile Communication (GS Mobicom) at Ilmenau University of Technology is communication in disaster scenarios. Due to a disaster or an accident, the network infrastructure can be damaged or even completely destroyed. However, available communication networks play a vital role during the rescue activities especially for the coordination of the rescue teams and for the communication between their members. Such a communication service can be provided by a Mobile Ad-Hoc Network (MANET). One of the typical problems of a MANET is network partitioning, when separate groups of nodes become isolated from each other. One possible solution for this problem is the placement of additional nodes in order to reconstruct the communication links between isolated network partitions. The primary goal of this work is the research and development of algorithms and methods for the placement of additional nodes. The focus of this research lies on the investigation of distributed algorithms for the placement of additional nodes, which use only the information from the nodes' local environment and thus form a self-organizing system. However, during the usage specifics of the system in a disaster scenario, global information about the topology of the network to be recovered can be known or collected in advance. In this case, it is of course reasonable to use this information in order to calculate the placement positions more precisely. The work provides the description, the implementation details and the evaluation of a self-organizing system which is able to recover from network partitioning in both situations.

Zusammenfassung

Das Hauptforschungsgebiet des Graduiertenkollegs “International Graduate School on Mobile Communication” (GS Mobicom) der Technischen Universität Ilmenau ist die Kommunikation in Katastrophenszenarien. Wegen eines Desasters oder einer Katastrophe können die terrestrischen Elementen der Infrastruktur eines Kommunikationsnetzwerks beschädigt oder komplett zerstört werden. Dennoch spielen verfügbare Kommunikationsnetze eine sehr wichtige Rolle während der Rettungsmaßnahmen, besonders für die Koordinierung der Rettungstruppen und für die Kommunikation zwischen ihren Mitgliedern. Ein solcher Service kann durch ein mobiles Ad-Hoc-Netzwerk (MANET) zur Verfügung gestellt werden. Ein typisches Problem der MANETs ist Netzwerkpartitionierung, welche zur Isolation von verschiedenen Knotengruppen führt. Eine mögliche Lösung dieses Problems ist die Positionierung von zusätzlichen Knoten, welche die Verbindung zwischen den isolierten Partitionen wiederherstellen können. Hauptziele dieser Arbeit sind die Recherche und die Entwicklung von Algorithmen und Methoden zur Positionierung der zusätzlichen Knoten. Der Fokus der Recherche liegt auf Untersuchung der verteilten Algorithmen zur Bestimmung der Positionen für die zusätzlichen Knoten. Die verteilten Algorithmen benutzen nur die Information, welche in einer lokalen Umgebung eines Knotens verfügbar ist, und dadurch entsteht ein selbstorganisierendes System. Jedoch wird das gesamte Netzwerk hier vor allem innerhalb eines ganz speziellen Szenarios – Katastrophenszenario – betrachtet. In einer solchen Situation kann die Information über die Topologie des zu reparierenden Netzwerks im Voraus erfasst werden und soll, natürlich, für die Wiederherstellung mitbenutzt werden. Dank der eventuell verfügbaren zusätzlichen Information können die Positionen für die zusätzlichen Knoten genauer ermittelt werden. Die Arbeit umfasst eine Beschreibung, Implementierungsdetails und eine Evaluierung eines selbstorganisierendes Systems, welche die Netzwerkwiederherstellung in beiden Szenarien ermöglicht.

Contents

1 Introduction **1**
1.1 Motivation . . . 1
1.2 Goals and Requirements . . . 2
1.3 Thesis Structure . . . 4

2 Mobile Ad-Hoc Networks **5**
2.1 Wireless Communication Networks . . . 5
2.2 MANETs – Common Description . . . 6
2.3 Typical Issues . . . 7
2.3.1 Self-Organization and -Configuration . . . 7
2.3.2 Power Restriction . . . 8
2.3.3 Restricted Capacities . . . 8
2.3.4 Dynamic Topology . . . 8
2.3.5 Lost Messages . . . 9
2.3.6 TCP and Dynamic Topology . . . 10
2.3.7 Security . . . 10
2.3.8 Partitioning . . . 10
2.4 Routing . . . 11
2.4.1 Basic Routing Algorithms . . . 11
2.4.2 Basic Types of Routing Protocols . . . 12
2.4.3 AODV Routing . . . 16
2.5 Summary . . . 16

3 Communication in Disaster Scenarios **19**
3.1 Disaster Management . . . 19
3.2 Communication Systems for Disaster Scenario . . . 21
3.3 Common System Requirements . . . 22
3.4 State of the Art . . . 22
3.4.1 Systems Using Additional Nodes Placement . . . 23
3.4.2 Systems Using Supporting Networks . . . 24
3.4.3 Systems Using Nodes Rearrangement . . . 24
3.4.4 System Using Message Ferrying . . . 26
3.4.5 Comparison and Evaluation . . . 26
3.5 Summary . . . 30

4 System Concept and Architecture 33
4.1 System Description and Goals . . . 33
4.1.1 Prerequirements and Main Assumptions . . . 33
4.1.2 Functional Requirements . . . 34
4.1.3 Non-Functional Requirements . . . 36
4.1.4 Communication Planes for Partitioning Recovery . . . 36
4.2 Availability of Global Knowledge . . . 37
4.3 Solution Methodology . . . 38
4.4 System Interfaces and Integration . . . 39
4.5 Unit Graph Model . . . 41
4.6 Use Cases . . . 42
4.6.1 Partitioning Recovery . . . 42
4.6.2 Coverage Improvement . . . 42
4.7 Summary . . . 42

5 Architecture Blocks 43
5.1 Click Modular Router . . . 43
5.1.1 Common Description . . . 43
5.1.2 AODV in Click . . . 45
5.2 Problem Detection . . . 47
5.2.1 Existing Approaches . . . 47
5.2.1.1 Prediction Systems . . . 48
5.2.1.2 Detection Systems . . . 48
5.2.1.3 ELFN . . . 49
5.2.2 Goals and Requirements . . . 50
5.2.3 Communication Partner Tracking . . . 51
5.3 Beaconing . . . 53
5.4 Location System . . . 54
5.4.1 Common Approaches for Node Location . . . 54
5.4.2 Outdoor Location Systems . . . 56
5.4.2.1 Coordinate system . . . 57
5.4.2.2 Distances Calculation Optimization . . . 57
5.4.3 GPS in Click . . . 59
5.5 Extended IP Headers . . . 60
5.5.1 Integration into IPv4 . . . 60
5.5.2 Integration into IPv6 . . . 61
5.5.3 Alternative Implementations . . . 63
5.5.4 Click Implementation . . . 64
5.6 Nearest Nodes Search . . . 64
5.6.1 Geographic-Aware Routing and Geocasting . . . 65
5.6.2 Greedy Perimeter Stateless Routing . . . 75
5.6.3 Heuristic Search . . . 76
5.6.4 Distributed Depth-First Search . . . 78
5.6.5 Problem Specific Extensions . . . 78
5.6.6 Comparison and Selection of the Different Algorithm . . . 79

5.6.7 NNS Implementation in Click . . . 80
5.7 Signaling and Communication Link (SCL) . . . 82
5.7.1 Design and Structure . . . 83
5.7.2 Data Access Module . . . 84
5.8 Extended Voronoi Diagram . . . 85
5.8.1 Voronoi Diagrams . . . 86
5.8.2 Construction Algorithms . . . 87
5.8.3 Algorithm Extension . . . 88
5.8.4 Implementation Details . . . 88
5.9 Nodes Placement Calculation . . . 89
5.10 Comparison of the Placement Scenarios . . . 91
5.11 Mission Requesting System . . . 92
5.11.1 Mission Requesting without Available Global Knowledge . . . 92
5.11.2 Mission Requesting with Available Global Knowledge . . . 93
5.12 Aggregation . . . 94
5.13 Summary . . . 96

6 System Evaluation 97
6.1 System Simulation . . . 97
6.1.1 Network Simulators . . . 98
6.1.1.1 NS-2 . . . 98
6.1.1.2 NS-3 . . . 100
6.1.1.3 OMNeT++ . . . 101
6.1.1.4 Evaluation . . . 104
6.1.2 Simulations . . . 106
6.2 Emulation in a Testbed . . . 106
6.2.1 Testbed Architecture . . . 107
6.2.2 Testbed Network Structure . . . 108
6.3 Implementation for Demonstrator . . . 111
6.3.1 Network . . . 112
6.3.2 Multicopter . . . 113
6.3.3 Server . . . 115
6.4 Summary . . . 116

7 Summary 117
7.1 Unique Benefits of the System . . . 118
7.2 Future Work . . . 118

Bibliography 123

List of Figures 139

List of Tables 141

List of Abbreviations 143

1 Introduction

The objective of this thesis is the definition of a system for partitioning recovery in mobile wireless networks with focus on communications in disaster scenarios. Network partitioning is a very common problem in communication networks, especially in mobile environment when some groups of the network participants may become isolated from the remaining part of the network due to their mobility, environmental effects, or damages of the network infrastructure caused by a disaster. The goal of this work is the development of technical algorithms and methods aimed to recover the network partitioning on a short-term base, and first of all to supply emergency communications.

1.1 Motivation

This thesis reflects one of the research topics within the International Graduate School on Mobile Communications (GS Mobicom) at Ilmenau University of Technology. The GS Mobicom is an inter-faculties organization with research focus on communications in disaster scenarios. The graduate school is active in the following topics related to this work:

- **Multicopter as a mobile communication platform.** The topic includes research on autonomous flight control and coordination of multiple multicopters.
- **Robust networking.** The theme consists of several PhD topics and investigates questions of routing and name resolution in mobile wireless networks.
- **Network nodes localization.** In this theme, the problems of in- and outdoor localization are studied in order to achieve a better localization accuracy and performance.

Along with the topics listed above, the graduate school also carries out research in the field of cognitive radio [LCLM11] and antenna design.

In a disaster scenario, partitioning of a (mobile) wireless network is a very common problem. Not only the nodes' mobility or failures can lead to partitioning but also the damages of terrestrial infrastructure of infrastructure-based networks. Some base station of cellular networks or some wireless access point can be damaged or even completely destroyed by a disaster, and due to this the network coverage area can

get some gaps. However, working communication is still very important for the areas affected by the disaster, first of all for emergency communications like SOS signals or the communications within and coordination of rescue teams. An example of a partitioned network is shown in figure 1.1: two network partitions A and B are reconnected by a multicopter using multi-hop communication.

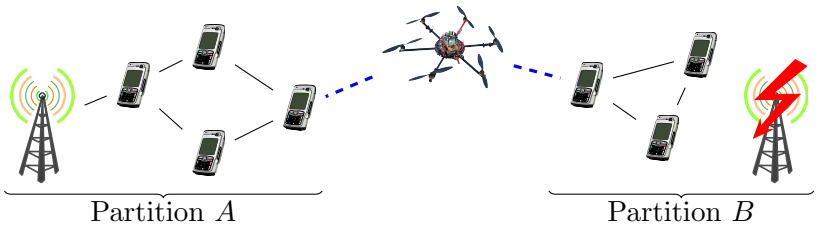


Figure 1.1: Network partitioning recovery after a base station failure

In such a situation, some additional technical resources are available for rescue teams. In the context of this work, the most interesting resources are mobile free movable devices like ground robots or Unmanned Aerial Vehicles (UAVs), which are capable of supporting communication functions by playing the role of message relays. The main idea of this work is to use these devices in order to reconnect any partitioned wireless networks.

At the same time, more and more people have their own mobile devices supporting different kinds of wireless communications – smart phones, tablet computers, netbooks, laptops etc. Since these devices are portable, there is a high probability that one of these devices will still be available for an injured person and can be used to request help. Furthermore, these devices provide additional means for the localization of victims. Even if a device is not equipped with a Global Positioning System (GPS) receiver, it can still be localized based on transmitted radio signals for wireless communications. Thus, providing communication possibilities for these "ordinary" portable devices is very important, too.

1.2 Goals and Requirements

A number of systems with the same goal is presented in the literature. An overview on these systems will be given in section 3.2. However, these systems have some drawbacks which will be explained in the corresponding section. Here, the goals and requirements for a network partitioning recovery system will be defined and discussed in detail.

Since the system is designed to support communications in disaster scenarios, it should be able to work with a possibly wide spectrum of wireless networking technologies. The system is intended to be independent of routing protocols used in the network which should be repaired, the only requirement to the routing mechanism is that the

routing can find a path from source to destination whenever a multi-hop communication between the source and the destination is possible physically.

One of the goal network types for the system is a Mobile Ad-hoc Network (MANET). MANET is a self-organizing group of mobile wireless nodes without any infrastructure and central controlling entity. Therefore, the system should provide self-organizing capabilities and be able to detect and to restore the network partitioning based only on the locally available knowledge using distributed control mechanisms. Of course, if a central control element is available in the network and aware of the actual network structure, the system should be able to use this in order to improve the performance and the accuracy of the calculation of the placement position for additional nodes.

Anyway, to be able to work, the system needs to know the positions of the nodes belonging to the network to be repaired. The location information can be provided either by a central localization system, if it available in the network, or by each individual node, which knows its own position or can estimate it.

Two different approaches to recover from network partitioning are possible: proactive and reactive. A proactive approach aims to prevent probable network partitioning using some partitioning prediction technique. The partitioning can be prevented by changing the network topology or by redistributing network resources, for example, by choosing an alternative route in order to minimize the total network power consumption, or to discharge some nodes. A reactive approach is attended to recover the network partitioning if it already took place. In this case, a reactive network partitioning recovery system is inactive until the network becomes partitioned and performs any activities only as response on the detected problem. Since the solution idea behind the system presented in this work is using additional nodes, which should be placed to reconnect isolated network partitions or to extend the network coverage of an infrastructure-based wireless network, the system should have reactive behavior. Otherwise, positioning of additional nodes for the purpose of preventing the partitioning needs too much energy even if the network will not be partitioned at all.

According to the main application scenarios, it is not allowed to change the positions of the network participants in order to prevent and/or to recover a partitioning. It is not possible to require from a member of rescue team to change his location only to provide network connectivity, since this will interfere with his primary task of life saving or of the disaster affected area restoration. Similarly, it is also impossible to ask a disaster's victim to change his position.

Additionally, the system should generate minimum communication overhead in order to prevent the resources from being exhausted on mobile devices, for which, first of all, battery power is a very constrained resource. Furthermore, the system should stay tolerant to single failures caused by nodes, whose batteries are flat.

A system design compliant to the named requirements is presented in this work. The presented network partitioning recovery system has a reactive behavior and supports both

self-organizing and centrally controlled design patterns, depending on the availability of global information about the network to be repaired.

1.3 Thesis Structure

The thesis structure reflects the goals mentioned above.

In the next chapter 2, an overview on wireless networks and especially on mobile ad-hoc networks will be given in order to provide the background information about the techniques considered in this work. Also, the chapter contains some information about routing protocols supporting multi-hop communications in mobile environment.

Chapter 3 includes the description of the problem specific to communications in disaster scenarios and a common view on a disaster management system. Furthermore, the chapter covers the state of the art analysis for existing partitioning recovery systems and their evaluation with respect to the aims of this work.

The next chapter 4 introduces the system concept and describes the structure of the whole system. In this chapter, the system is presented as a set of interconnected components which interact with each other. The interfaces of the system and several usage scenarios will be discussed in this chapter, too.

Chapter 5 goes deeper into system realization and describes the implementation details for each separate component and the algorithms used by these. As pre-required information, the chapter also provides a short overview of the Click software framework, used for implementation.

The developed system is evaluated in chapter 6. Three evaluation phases are described in this chapter: simulation of individual system components and parts of the system, emulation of the system in an artificial environment and tests on the joint network recovery demonstrator.

Finally, chapter 7 concludes the work. In this chapter, an overview of the developed system will be given, with respect to existing systems, presented in chapter 3. The system constrains and limitations will also be discussed there together with possible directions of future work.

2 Mobile Ad-Hoc Networks

In this chapter, a short overview of wireless communication networks will be given with focus on Mobile Ad-hoc Networks (MANETs).

2.1 Wireless Communication Networks

Communication networks can be divided into two big classes: wired and wireless networks. Nowadays, wireless networks have a very wide range of applications and they are also an important topic in scientific world. Studying MANETs requires an interdisciplinary approach since various scientific areas are involved here: physics of radio signal propagation, signal processing and coding, topology graph analysis, distributed calculations and algorithms, self-organization and -configuration and many others.

There are many ways to categorize the wireless networks. The most important of them are presented in [LC05]. First of all, the wireless networks can be infrastructure-based and infrastructureless. Typical examples of infrastructure-based wireless networks are mobile phone networks, like GSM and UMTS, since they need an established terrestrial infrastructure consisting of base stations and other controlling components. Satellite communication networks and Wireless Local Area Networks (WLAN) belong also to the infrastructure-based networks. Examples of the infrastructureless networks are a Bluetooth network of two or more mobile devices like smart phones, an established Infra-Red (IR) link or a WiFi network in ad-hoc mode.

Secondly, the wireless networks can be classified by their communication area as shown in fig. 2.1:

- **WAN – Wide Area Networks**. The networks with large coverage area of cities or even whole countries. Common examples of these wireless networks are cellular mobile phone networks like GSM and UMTS. These networks are also infrastructure-based.
- **MAN – Metropolitan Area Networks**. These networks can cover for example a university campus area, or provide communication between several buildings when a wired or fiberglass link is not possible or does not suit because of some reason. These networks also use an established infrastructure like a set of wireless Access Points (AP) for a WiFi-based network.

- **LAN – Local Area Networks**. The networks with coverage range within some hundreds meters. It can be a WiFi hot spot, or a WiFi network within an office or room. These networks can be both infrastructure-base and -less.

- **PAN – Personal Area Networks**. Typically, it is a network of several mobile devices connected to each other without any infrastructure elements. The communication range of such a network is about tens of meters. Basic technologies are WiFi in ad-hoc mode, Bluetooth and Infrared (IR).

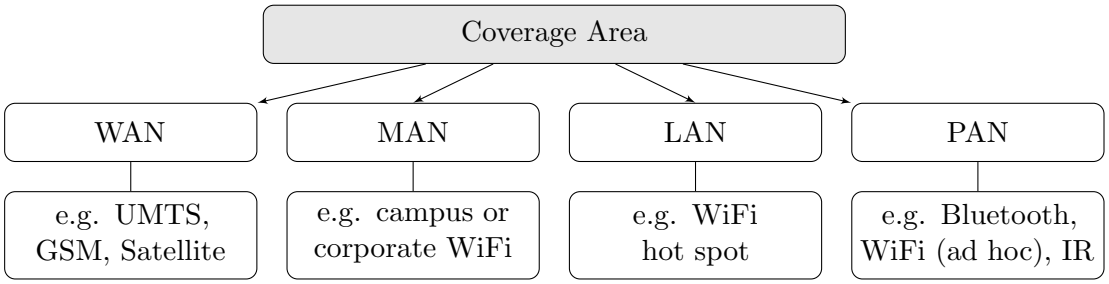


Figure 2.1: Wireless networks by communication area, according to [LC05]

Additionally, wireless networks can be categorized according to their application: enterprise, home, tactical, sensor, pervasive, wearable, and automated vehicle networks.

All the networks can be combined together using nodes with multiple different network interfaces - gateways. The gateway is a border node between networks of different types. For the upper layers of a communication protocols stack, the gateway should provide transparent communication, as if a packet is transmitted within only one network. Such a type of the network structure is called hybrid network.

One important class of wireless networks is built by Mobile Ad-hoc Networks (MANETs). MANET is a self-organizing wireless infrastructureless network of free movable mobile nodes. The communication range of MANETs can be between tens and hundreds of meters. These networks are being actively studied in the last decades because they provide a great interest from practical and scientific point of view.

In the following sections, the MANETs will be discussed in more detail.

2.2 MANETs – Common Description

As mentioned before, a MANET is a wireless network without infrastructure. As a rule, the MANET does not have any central controlling element, and as consequence, the nodes should be able to solve the main networking tasks, like routing, addressing and possibly name resolution in a distributed manner.

Thus the MANET is a self-organized system, where each node works using only information about its local environment according to predefined rules. As follows from the self-organization, all the algorithms used in the MANET should also be distributed.

As a self-organizing system, a MANET is robust against changes within its structure: the nodes can leave or join the network at any time. Since the nodes need only local awareness about the network, the system is also scalable against the number of nodes participating in the network.

A distinctive feature of MANETs is multihop communication (e.g. fig. 1.1), so each node in the network provides router functionality and can forward messages to other nodes. From this point of view, all the nodes of the network are equal and can be substituted by each other. On the other hand, MANETs are often heterogeneous and utilize simultaneously several networking techniques. For example, one part of the nodes uses Bluetooth connection and the other WiFi connection. In such a case, the nodes equipped with both interfaces can provide also a gateway functionality between both parts of the network.

2.3 Typical Issues

From the nature of things, the following issues are typical for MANETs, which present different research directions and should be taken into account when developing algorithms used there.

2.3.1 Self-Organization and -Configuration

Since a MANET is a self-organizing and self-configuring system, there is typically no central entity which would have a coordination function. Thus, all nodes in a MANET are equal in terms of controlling functionality.

Scott Camazine gives the following definition of a self-organizing system in [Cam03]:

> “ *Self-organizing systems are physical and biological systems in which pattern and structure at the global level arises solely from interactions among the lower-level components of the system. The rules specifying interactions among the system's components are executed using only local information, without reference to the global pattern.* ”

In other words, global behavior of a self-organizing system is a product of local interactions of its particular elements. Since there is no central controlling entity in the system, all algorithms utilized there should be distributed. Using only the local

information, the system should be able to achieve some stable desired state according to purposes of the whole system. In case of self-organizing communication networks, such a network should be able to deliver an information from point A to point B without the knowledge about the whole structure of the network.

As members of a self-organizing system, the nodes in MANET observe their local environment and act according to predefined rules using only this locally available knowledge. Also a MANET is robust against nodes' arrivals and departures, so the network will be still existing while there are at least two nodes which can communicate with each other. Of course, the network capacity can deteriorate or even some services can fail, if some important nodes leave the network, but the communication within the network will be still possible.

2.3.2 Power Restriction

As MANETs usually consist of mobile nodes, one of the major problems for them is power supply. Typically, the smaller a device is and the higher its computational capabilities, the more energy it consumes. A considerable part of energy is expended by wireless communications. Thereby, the lifetime of the nodes without extra charge can be about only a few hours.

In turn, it leads to restrictions for the software and communication protocols used in MANETs. So, the longer an application can be in sleep or in hibernated mode, the longer the node can stay alive. Similarly, the less messages are required to be transmited by the node, the longer the battery's energy will suffice.

2.3.3 Restricted Capacities

Another feature of mobile nodes is the limitation of memory capacity and computational power of the nodes. Though memory capacities and computational power have grown, and memory and processors become cheaper continuously, they still can be a limitation factor of the MANET's usage. Furthermore, high computational power leads to high energy consumption, which, as mentioned before, is still a big problem of MANETs.

2.3.4 Dynamic Topology

Another major issue of MANETs is the nodes' mobility and, as consequence, a dynamically changing topology of the network. To overcome this issue, an efficient routing mechanism should be available in the network. The routing issues will be considered in the following section 2.4 in more detail. If the network provides some services, these should be deployed redundantly in case some nodes will be temporally unavailable.

2.3.5 Lost Messages

During the constantly changing topology and environmental effects of wireless communication, messages in a MANET can be lost often and should be retransmitted again. As consequence, the network capacity decreases significantly. Another reason for packet loss is the multiple medium access that often leads to collisions.

Additionally for infrastructureless wireless network, the problems of hidden- and exposed-terminal are well known [JPDG04]. Both problems are shown in fig. 2.2. On the left side, the hidden-terminal problem is illustrated. Nodes A and C have a data to be transmitted to node B. Node A senses the medium, detects that it is free, and starts transmission. Simultaneously, node C senses the medium, does not detect any transmission from node A because it is out of the range of node A, and starts transmission, too. Both transmissions collide at node B and fail.

At the right side of fig. 2.2, the exposed-terminal problem is shown. In this case, node B has data for node A, and node C has data for node D. Theoretically, both transmissions can be done simultaneously because they will not interfere at the end nodes. However, if node B starts the transmission first, node C cannot transmit because from its point of view the medium is occupied.

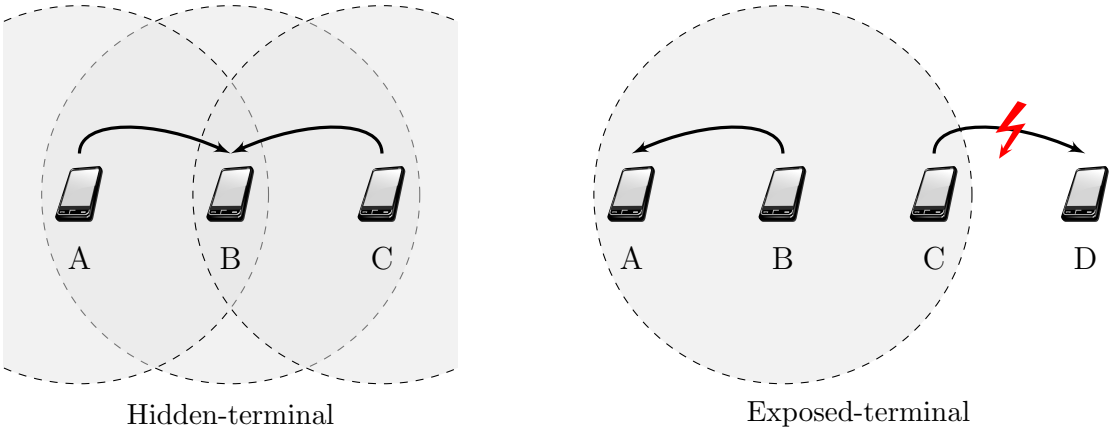


Figure 2.2: Hidden- and exposed-terminal problems

Typically, both problems are solved using the RTS/CTS mechanism (Request To Send / Clear To Send). If a node has data to be transmitted, it emits at first a small RTS packet. The corresponding node answers with a CTS packet, and after that the data can be transmitted. Special handling rules for RTS and CTS packets make it possible to avoid collisions, but it costs some additional communication overhead and delays.

2.3.6 TCP and Dynamic Topology

Because of the highly dynamic topology, TCP [RFC81] connections often lose their efficiency in MANETs. Continuously changing routes, emergence or disappearance of links lead to high packet loss rate. In this way, if an acknowledgment packet (ACK) is lost, TCP performs retransmission of the data packet, even when the data packet itself was received successfully. Missing ACKs can be also interpreted as a signal of congestion by the TCP congestion control mechanism, and the transmission rate will be slowed down even more. All this leads to a significant decreasing of TCP connection throughput or even to link breakages.

In the literature, numerous solutions are proposed and investigated. Some of them describe additional mechanisms for the existing TCP protocol, others propose completely different alternative protocols. For example, in [HV02] an Explicit Link Failure Notification (ELFN) mechanism is used to notify the TCP source node about the link failure. According to this notification, the source node performs appropriate actions within the TCP congestion control mechanism.

However, the detailed description of TCP problems in mobile environment is out of the scope of this work.

2.3.7 Security

Because of the fact that MANETs are typically public spontaneously emerging networks, everyone can relative simply participate. Furthermore, during dynamical routing each communication packet can traverse potentially over every node and can be intercepted by them. Thus, the securing of communication it is a big and very important issue of MANETs.

However, the security issues are out of the scope of this work, since this problem can be an objective of separate work and is a well researched topic in computer science, too.

2.3.8 Partitioning

The common problem of MANETs is network partitioning. Because of highly dynamic topology, nodes and links failures, it is possible, that some groups of nodes become isolated from others. Within each group, communication is still possible, but there are no connections between groups.

In spite of the fact that this problem can take place very often, its solution is a quite difficult task and even not ever possible at all. The transmission power of nodes is physically limited, signal attenuation is caused by the nature of electromagnetic

waves. As consequence, the nodes' communication range is hard limited. Thus, the reconnection of isolated partitions is only possible by changing the network topology (positions of nodes) or by placement of one or more additional nodes into the gap between partitions.

Both solutions can only be applied in a limited way, because it is not always possible to change the positions of nodes in a MANET intentionally, and the additional nodes can be also not available in the network. However, the main objective of this work is the network partitioning reconnection using the placement of additional nodes.

2.4 Routing

As mentioned above, the Routing in MANETs is a special challenge because of the highly dynamic topology of such networks. For this reason, this section provides a more detailed description of routing mechanisms used by MANETs. In this section, materials of a student's advanced seminar work "MANET-Routing: Klassifizierung und Vergleich von Routing-Protokollen" (engl. "MANET Routing: Classification and Comparison of Routing Protocols") [Hen10b] are used.

2.4.1 Basic Routing Algorithms

Because a MANET topology can been seen as a topology graph, the base algorithms for routing can be adopted from the graph theory.

Link State Routing For purposes of link state routing, the nodes disseminate information about themselves and their neighbors. Thereby, the nodes are aware about their own position within the network and can calculate an optimal route using a shortest path search algorithm. Actual network state information is distributed within link state announcements, and the links between neighbor nodes are monitored with special control packets. Certainly, the control information exchange takes away a certain amount of bandwidth and energy [PGC00].

The most famous shortest path search algorithm is Dijkstra's algorithm [Dij59]. The algorithm belongs to the group of greedy algorithms, and starts the search with the path promising the best results. For the shortest path search, the edges of the graph (network links) get a weight corresponding to the length according to some metric (hop count, link quality, bandwidth, etc.). The Dijkstra's algorithm follows the shortest path at first, and only after that the longer paths will be investigated. The algorithm requires the knowledge about network topology and link costs (lengths, weights).

Another popular base algorithm for the routing is A* algorithm [RN09]. It extends Dijkstra's algorithm using heuristics to achieve better performance. The A* algorithm makes it possible to utilize not only properties of the network graph but also some additional information about the environment, so it can, for example, avoid some unwanted links during the routing process.

Distance Vector Algorithm Anther major class of routing protocols bases on distance-vector routing algorithm. The distance-vector routing utilizes the Bellman-Ford algorithm [Bel58, CLR09] or the Ford-Fulkerson algorithm [FF56] for path finding. In opposite to the link state routing, the nodes should inform only their neighbor about the topology changes, and not all nodes of the network. So the communication overhead and computational complexity can be reduced significantly.

The nodes should not have knowledge about the entire path to the destination, but only a direction and a distance, like a vector, towards the destination. The direction is given by the interface or by the neighbor towards the destination, and the distance is typically a hop count to the destination. However, some other parameters like link travel time or link capacity can be utilized as the distance parameter.

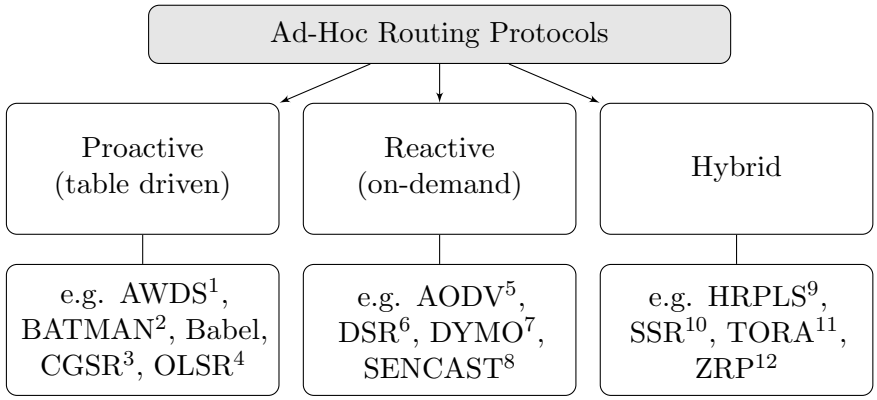
2.4.2 Basic Types of Routing Protocols

There many possibilities to categorize MANET routing protocols. The most common way is to divide the protocols into proactive, reactive and hybrid protocols as shown in fig. 2.3. The authors of [FLS06] propose to classify the routing protocol, according to the destination count on unicast, multicast and broadcast.

Proactive Routing Proactive or table-driven routing collects the routing information within the whole time of network's life. Thus, the route is already known when a data packet should be transmitted. The advantage of the proactive routing scheme is low delay when sending data . However, the continual exchange of routing information leads to additional load of the network and also increases energy consumption.

Furthermore, during proactive routing paths will be established which are never used. Also, a big network reacts on the topology changes, arrivals and departures of nodes slowly, and because of this the proactive routing is not really suitable for networks with highly mobile nodes.

Some examples of proactive routing protocols are shown in table 2.1. In the table it is also denoted which basis routing algorithm is utilized by each protocol: Link-State (LS) or Distance-Vector (DV). Additionally, some of the routing protocols support multicast and Quality-of-Service (QoS).



[1] **A**d-hoc **W**ireless **D**istribution **S**ervice
[2] **B**etter **A**pproach **T**o **M**obile **A**d-hoc **N**etworking
[3] **C**lusterhead **G**ateway **S**witch **R**outing
[4] **O**ptimized **L**ink-**S**tate **R**outing
[5] **A**d-hoc **O**n-demand **D**istance **V**ector Routing
[6] **D**ynamic **S**ource **R**outing
[7] **Dy**namic MANET **O**n-demand Routing
[8] **S**calable Protocol for Large Ad-hoc **E**mergency **N**etwork for Uni**cast**ing and Multi**cast**ing
[9] **H**ybrid **R**outing **P**rotocol for **L**arge **S**cale MANETs with Mobile Backbones
[10] **S**calable **S**ource **R**outing
[11] **T**emporally-**O**rdered **R**outing **A**lgorithm
[12] **Z**one **R**outing **P**rotocol

Figure 2.3: MANET routing algorithms

Reactive Routing In opposite to proactive routing, reactive routing initiates a route discovery firstly when a data packet should be transmitted. Thus, the data transmission along a new found route can only be started after some delay needed for route discovery. This makes it possible to have the actual route any time it is needed. Because of on-demand route construction, the energy consumption of nodes is also lower than in case of proactive routing when there are no data to be transmitted.

However, the network is often flooded with route requests, and it can lead to congestions if more nodes start their route discoveries simultaneously.

The example of corresponding routing protocols are shown in table 2.2. Due to the special importance for this work, the Ad-hoc On-demand Distance Vector (AODV, [PRD03]) routing protocol will be described below in section 2.4.3 in more detail.

Hybrid Routing Hybrid routing protocols try to combine the advantages of both reactive and proactive routing. In this case, different network parts can utilize different routing schemes. For example, the nodes maintain the routing tables for proactive routing only for the nodes within some radius (in terms of hop count), and use a reactive routing protocol to communication with the nodes which are farther away. Of course, a

Table 2.1: Proactive routing protocols examples

Routing Algorithm	Basis	Multicast Support	QoS Support	Source
AWDS	LS	no	no	[L+07]
BATMAN	LS	no	no	[QLL]
Babel	DV	no	no	[Chr11]
CGSR	DV	yes	no	[HKCW+97]
OLSR	LS	yes*	yes*	[CJ03]

* – with extensions

Table 2.2: Reactive routing protocols examples

Routing Algorithm	Basis	Multicast Support	QoS Support	Source
AODV	DV	yes	yes	[PRD03]
CHAMP	n.a.	no	no	[VSR03]
DSR	n.a.	yes	yes	[JM96, JHM07]
DYMO	n.a.	yes*	yes*	[PC12]
SENCAST	DV	yes	yes	[AK08]

* – with extensions

switching mechanism is needed to decide which type of routing should be used at any point in time.

This idea is used in Zone Routing Protocol (ZRP) [HPS02] where each node maintains a routing table for proactive routing for all neighbors within a ρ-hop zone. For the nodes out of this zone, a route discovery mechanism is utilized according the reactive model. Other examples of the hybrid routing protocols are shown in table 2.3.

Other Routing Algorithms There is a number of other protocols which are aimed to solve a specific task, or optimized with respect to one specific parameter. Nodes in a MANET are often equipped with some navigation device or can detect their own position using a location system available in the network. Therefore, location-aware routing protocols are very popular in research. Using a geographic routing protocol makes it possible to access nodes over their geographical coordinates, or even a group of nodes within a defined geographical area (geo-multicast or geocast). Because one of these geographic routing protocols is used as a base algorithm in this work, geographical routing and geocasting will be considered separately in section 5.6.1 “Geographic-Aware Routing and Geocasting” in more detail.

Table 2.3: Hybrid routing protocols examples

Routing Algorithm	**Basis**	**Multicast Support**	**QoS Support**	**Source**
HRPLS	DV	no	yes	[PAKG06]
SSR	own	no	no	[FDKC06]
TORA	LR*	no	yes	[PC01]
ZRP	LS & DV	yes	yes	[HPS02]

* – **L**ink **R**eversal routing algorithm [BST03]

Many other routing protocols which are not relevant for this work are listed below to keep the overview of MANET routing protocols complete. The Flow-Oriented Routing Protocol (FORP) [SLG01] takes existing data flows into account in order to find routes with a requested capacity. The Hierarchical Core Extraction Distributed Ad hoc Routing (CEDAR) [SSB98] splits the network nodes into clusters so that different nodes have different tasks: regular nodes should be able to communicate only with their cluster head, and the cluster heads are responsible for the inter-cluster communication. Some protocols are aimed directly to support multicast transmission, for example the Protocol for Unified Multicasting Through Announcements (PUMA) [VGLA04], to address groups of nodes.

Other routing protocols try to optimize the routing approach to reduce the consumed energy, to keep the routing secure or to provide some QoS capabilities. Examples are the Power-aware Source Routing (PSR) [MDP02], the Security-Aware Routing (SAR) [YNK02] and QoS-aware routing[AZM05] respectively. Context sensitive routing proposed in [Deb09] tries to combine mechanism for searching of services within the network with routing to provide a user with an optimal service within the current user context.

Another more sophisticated routing approach is presented by component based routing [LZH+06] where the routing mechanism is provided by a combination of different components. Combining this components, a special routing mechanism can be established to provide the functionality needed in a specific situation.

Such a variety of routing protocols is caused by that many proposed protocols deal oft with solution for one specific problem only, without consideration of other problems. Many of these protocols along with certain advantages have drawbacks which make it impossible (or inconvenient, or inefficient) to use them in real MANETs. However, there are several protocols which are commonly used for MANETs and which are implemented for a number of network simulators and for real platforms like Linux. To these protocols belongs for example the Ad-hoc On-demand Distance Vector (AODV) routing which is discussed below.

2.4.3 AODV Routing

AODV [PRD03] is a well investigated approach for routing in MANETs. It is supported by most major network simulators and also several AODV implementations are available for a Linux-based platform. As a simple ad-hoc routing protocol, AODV is used as base routing mechanism for simulations, tests and validation within this work. Therefore, it will be considered here in more detail.

AODV is a reactive routing protocol based on the distance-vector algorithm, it also supports both unicast and multicast routing. AODV utilizes route sequence numbers during the route discovery to avoid the "count-to-infinity" [KU10] problem.

In AODV, data transmission is not active (except `HELLO` messages) if there is no necessity to discovery or to repair a route. If a network node has data which should be transmitted, it initiates a route discovery process by sending out a broadcast route request (`RREQ`). Other nodes getting the route request, retransmit it again and add a record in their routing tables with the route to the node that generated the `RREQ`. If node already knows a route to the destination in the route request, it either answers with a route reply message (`RREP`) or forwards the route request to the destination if the "destination only" flag `D` is set.

During the routing discovery process, only the routes with the minimum hop-count towards the destination are used. Unused routing table records are firstly invalidated and then deleted after some pre-defined timeouts. If for some reasons a route was broken, it is indicated by a route error message (`RERR`) and the route discovery process is started again.

There are also several mechanisms aimed to reduce the communication overhead during the routing process. Firstly, the route requests are aggregated according to the destination sequence numbers. Secondly, the path length can be restricted with some given number of intermediate hops. Furthermore, a binary exponential backoff mechanism is utilized when a route request failed. It means, the timeouts between route request retransmissions are doubled after each retransmission.

The advantage of AODV is that it does not generate any additional traffic by the data transmission along an already established path due to updating of route life time in routing tables of the nodes participating in the data transmission. However, it needs more time before the first data packet can be sent out because of the route discovery process, as opposite to proactive approaches.

2.5 Summary

Summarized, an overview on wireless networking technologies was given in this chapter with focus on techniques which corresponds to the goals of this work. Firstly, mobile

ad-hoc networks were discussed here with respect to the typical problems, which should be solved during the usage of MANETs.

Secondly, the problems of routing in MANETs were presented here with focus on AODV routing, which is chosen as the basis routing mechanism within the scope of this work.

The following chapter continues the discussion of the problems relevant to the system developed within this work and deals with specific of MANETs' application in disaster scenarios. Also the following chapter provides an overview of existing network partitioning recovery systems with their comparison and evaluation in context of the goals of this work.

3 Communication in Disaster Scenarios

The main problem domain motivating this work is communications in disaster scenario. Along with the general features of MANETs, this scenario brings additional requirements and restrictions for the developed system. For a better understanding the specific of the communication scenario, a short overview of disaster management systems will be given in the first part of this chapter.

One of the typical problems of MANETs, as described in the previous chapter, is network partitioning. This problem is especially actual for communications in disaster scenarios: partially destroyed terrestrial network infrastructure leads to emerging isolated network parts and can be substituted with a MANET. Rescue teams consist typically of several groups which need to communicate with each other. The communication within a group can be possible due to short distances between the group's members, but different groups can be out of the range of each other. In this case, the groups build a partitioned MANET, which should be reconnected. The second part of this chapter provides information about existing systems for network partitioning recovery in MANETs and WSNs and discusses the main ideas behind them.

3.1 Disaster Management

In this section, common principles of disaster management are discussed. Note that due to specific of this work, only technical aspects of disaster management is considered here.

The main goals of disaster management are minimizing risks and damages caused by a disaster and supplying timely countermeasures. In literature, four phases of disaster management are defined [TI10]:

- **Mitigation** is a set of preventive measures aimed to reduce the risks and effects of a disaster before the disaster happens. In the background of the mitigation lies risk analysis which results in technical and organizational measures aimed to prevent hazards from developing into disaster or to minimize loss if the disaster is unavoidable.

- **Preparedness** is aimed to limit the impact of disaster events on people. This phase includes trainings, organization, equipping, monitoring and other activities to ensure the correct behavior and reaction in a real disaster situation. This phase also includes prediction of possible damages, injuries or deaths.

- **Response** is the phase following directly the first manifestations of a disaster. This phase includes the actions directed to rendering first aid to possible victims. It is the most critical phase, whose proper organization determines how many human lives can be saved.

- **Recovery** is a restoration phase, which comes after all victims, which can be rescued, are rescued. This phase aimed to recover the area affected by the disaster and its infrastructures.

The four phases build a closed disaster management cycle depicted in figure 3.1. The four phases concept includes proactive and reactive phases, which not only allow the elimination of disaster effects, but also hazards prediction and risks minimization before the disaster happens.

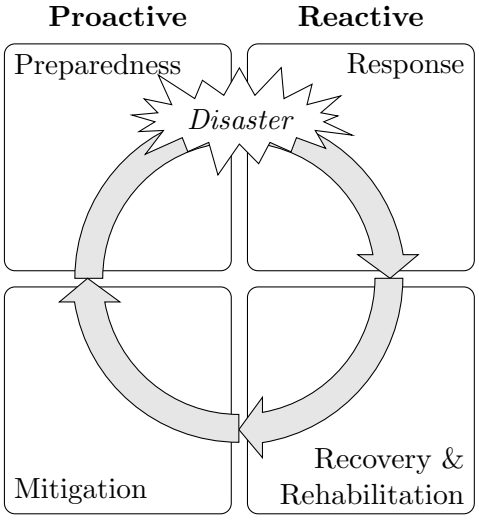


Figure 3.1: Disaster management cycle concept (adopted from [TI10])

In the response phase, technical and organizational means play a vital role. Proper organized communication directly affects the number of saved lives. Additionally, the communication systems play an important role in the preparedness phase e.g. for the early warning systems. In the next section, the usage of communication networks in disaster scenarios will be discussed in more detail.

3.2 Communication Systems for Disaster Scenario

During the response phase, communication networks can have several possible applications. First of all, they are used for coordination and communication of the rescue forces. The networks should be able to support voice, video and data communications and enable the communication between rescue teams and headquarters. Voice communication is used to coordinate the rescue operations, and data communication provides information needed for these activities (e.g. geographical maps, list of required and available facilities, etc.).

Nowadays, almost all peoples have a mobile telephone or a smart phone, which are able to connect to one or more wireless networks. In an emergency situation, such a smart device can be also used to request help. When the terrestrial infrastructure of cellular networks is damaged or completely destroyed, these devices are still able to build an ad-hoc network, and emergency information can propagate over this network. Since most of modern devices are equipped with a GPS receiver, they can help significantly the rescuers to find victims owning the devices. Using a smart mobile device, important information about the condition of the victim can be delivered to the rescue headquarter. This information can also be taken into account when planning the rescue operation.

In a disaster scenario, every available communication technology can and should be used. However in such a situation, the wireless infrastructureless or satellite communications have advantages over the wired communications, since the wireless networks typically have mobile end-devices which can still be available during or after the disaster. Because of this fact, the usage of MANETs in a disaster scenario is a very promising field in communication network research.

Since mobile devices typically have restricted resources, especially limited battery power, MANETs can provide only a short- and medium-term support for communications due to their short lifetime (maximal some hours). In spite of this, MANETs are still useful for emergency communications, when mobility is very important. Similarly as any other MANETs, the MANETs used in emergency situations have the typical problems like network partitioning, which should be solved efficiently. The remainder of this work deals with the development of a network partitioning recovery system for disaster scenarios.

The developed network partitioning recovery system is primarily destined to the response phase and supports a reaction to appearing communication problems. The system is not aimed to provide a long-term solution for the repairing or replacement of the destroyed network infrastructure. The main goal of the system is to provide an ad-hoc mechanism to reconnect network partitions in support of emergency communications.

Another important role for the communication system used in disaster scenarios plays their survivability and robustness to fails of some nodes. This can be achieved using a decentralized (or distributed) controlling mechanism instead of a central controlling entity, which can build single point of failure.

3.3 Common System Requirements

According to the usage scenarios, the system for network partitioning recovery should fulfill the following requirements:

- **Distributed controlling mechanism.** It provides the robustness of the system and protect it from problems caused by single point of failure.
- **Usage of additional devices.** The network participants should not be required to change their behavior for network restoration since it can block the main tasks, performed by the node.
- **Energy efficiency.** Since the primary goal of the developing system is MANETs, the system should take into account the power restrictions of mobile nodes and eliminate not necessary communications.
- **Reactive behavior.** This requirement is bound with the previous one and guarantees that the extra resources will not be consumed if it is not required at the moment.
- **Work with standard devices.** The developing system should be primarily designed to support communication between "ordinary" devices like smart phones and notebooks.

In the following section, some existing solutions for the network partitioning recovery will be discussed with respect to the named requirements and to the context considered in this work.

3.4 State of the Art

Network partitioning recovery can be split into two phases: the detection (or even prediction) of partitioning and their reconnection. The existing systems and methods for network partitioning detection only will be discussed later in section 5.2.1. In this section, an overview of the systems aimed to reconnect network partitions is provided. The materials of this section are partially based on a student's advanced seminar work "Erkennung und Behebung von Netzpartitionierungen in MANETs und WSNs" (engl. "Network Partitioning Detection and Recovery in MANETs and WSNs") [Sch12].

There are several ways to (re)establish the communication between isolated parts of a network. The most obvious solution is the placement of additional nodes between the network partitions (e.g. [DPS08, CDB04]). As an extrapolation of this idea, a whole additional network can be deployed in the area needed to be covered for communication (e.g. [DPH05, MCG09]). Other solutions propose to rearrange the network nodes in

order to reconnect them (e.g. [ASTU10, WL09]). An alternative for this is the usage of the Delay Tolerant Networks (DTNs) and message ferrying to transport messages between isolated groups (e.g. [WL10]). However, DTNs do not allow any real time data transfer what is needed for voice and video communications. Alternatively, there are methods which aimed to delay the network partitioning caused by the failed nodes due to the power limitation as long as possible using load balancing strategies. Primarily, such systems are invented for WSNs with not movable nodes (e.g. [Sim08]). All the named systems will be discussed in the following section in more details.

3.4.1 Systems Using Additional Nodes Placement

The authors of [DPS08] present a system for network partitioning recovery in WSNs with a static topology. The nodes use multi-hop routes to communicate with a base station. The nodes which can be reached from the base station form a *safe* partition, all other nodes belong to an *isolated* partition separated by a gap. The base station detects a partitioning using the Partitioning Detection System (PDS) from [SST05]. For this purpose, the base station randomly selects a small number of nodes, so called *sentinels*, and polls them periodically. If some *sentinels* do not answer, the base station detects that the partitioning happened and sends out a `DISCOVERY` message in order to find the borders of the safe partition. It is assumed that the base station knows the positions of the nodes, which can be predefined or estimated using a localization system. Since the borders were detected, the base station broadcasts a message with a new random *epoch* number, which is saved by each reachable node. So, the nodes in the safe and isolated partitions have different *epoch* numbers. To reconnect the partitions a movable mobile node is used. The mobile node is directed by the base station to the border of the safe partition. During the movement, the mobile node checks the *epoch* numbers of the neighbor nodes. One the one hand, it can recognize when it reaches the partition border if different *epoch* numbers were detected. On the other hand, the check of the *epoch* numbers can guarantee that the mobile node will not be disconnected from the safe partition. If necessary, more mobile nodes can be used to reconnect the partitions.

The aim of the system presented in [CDB04] is providing full connectivity for a ground MANET using Unmanned Aerial Vehicles (UAVs). The ground network consists of several isolated groups of nodes. The nodes within a group can communicate with each other but not with the nodes from other groups. The nodes belonging to one group build a cluster. The authors demonstrate a heuristic centralized algorithm aimed to determine the minimal number of UAVs needed to connect the clusters. The algorithm tries to find optimal or suboptimal placement positions for UAVs to build a fully connected network. As input, the algorithm takes the nodes connectivity matrix which is constructed based on communication ranges and terrain constrains. The algorithm uses nodes location history in order to predict the nodes movement, which is taken into account by calculating the placement positions for UAVs. UAVs and the ground

nodes together build a two-layer ad-hoc network: layer of the ground nodes and layer of UAVs. It is assumed that the ground nodes have two different network interfaces: one to communicate with each other, and one to communicate with UAVs. As a benefit of two interfaces, the authors indicate the possibility of providing different bandwidth capabilities on each interface. However, a routing protocol used in the network must support these multiple interfaces.

3.4.2 Systems Using Supporting Networks

Another interesting solution for providing network connectivity within large areas was proposed in [DPH05]. Instead of calculation of the optimal positions for the placement of individual additional nodes, a number of small nodes called microrouters is distributed uniformly over the area so that these nodes build a fully connected network. The only task of these nodes is multi-hop message forwarding from a source to a sink, where the source and the sink are the nodes of a partitioned MANET which needs the connectivity. In other words, the network of these small nodes forms an infrastructure (or supporting network) for the MANET nodes. The solution is very simple – it is enough that the both groups of nodes support a common routing protocol, but it is also very expensive due to a large number of the nodes which should be distributed over the area.

The next step in the similar direction was done in project *LANdroid* of the Defense Advanced Research Projects Agency (DARPA) [MCG09]. In this project, the supporting network is built by a number of robotized nodes called LANdroids. The robots construct the network with maximal coverage of the given area in self-organizing manner. The LANdroids are designed to operate in urban environment with a large number of obstacles disturbing radio signal propagation (no line-of-sight between the nodes). By the design, the network of LANdroids should be self-healing if some nodes leave the network because of lack of energy. Again, the network is very expensive due to the usage of a big number of mobile robots.

3.4.3 Systems Using Nodes Rearrangement

The authors of [ASTU10] proposed a distributed PArtition Detection and Recovery Algorithm (PADRA). In opposite to the previous systems, this solution works into another direction: no additional nodes are used, instead of that the existing network nodes are rearranged. The system demonstrates a proactive behavior. At the beginning, the network consists of randomly distributed mobile nodes and is connected. For the network, the following roles of nodes are defined: a *dominatee* node is a node which has only one neighbor (corresponds to a leaf in a tree topology). All other nodes are called *dominator*s which are potentially critical or *cut-vertex* nodes. The failure of a *cut-vertex* node can lead to network partitioning. Each node of the network is aware of its two-hops neighborhood. Firstly, each *dominator* has to detect whether

it is a *cut-vertex* or not. For this purpose, each neighbor of the *dominator* tries to find alternative routes to other *dominator*'s neighbors using a local depth-first search approach. If the alternative routes between all the neighbors were found the *dominator* is not a critical node. Secondly, a *dominator*, which has detected that it is a *cut-vertex*, should find one or more *error handler* nodes, which should change their positions in order to reconnect the network if the *cut-vertex* fails. Primary candidates to be *error handler*s are the *dominatee*s since they can be moved freely without the threat of network partitioning. In order to detect network partitioning, each *cut-vertex* node periodically sends beacon messages to its neighbors. If within a predefined time interval a neighbor does not receive any beacons, it initiates the recovery process. During the recovery, the corresponding *error handler* nodes rearrange their positions to substitute the failed node. In order to optimize energy consumption during the movement, the network can decide to rearrange more *error handler* nodes. In this case, each node should move a shorter distance, than it would be if only one *error handler* moved.

A similar idea of rearranging nodes is discussed in [WL09]. It presents an active node allocation scheme in which the nodes should change their positions in order to achieve a regular number of neighbors. Depending on the given parameters, each node performs a directed movement to reach new neighbors if it has to few neighbors or to reduce the number of neighbors if it has to many. Due to such a movement, the network can spread over a large area and provide an optimal coverage of this area with a stable connectivity. In order to orient oneself within the network, the nodes should save each movement (distance in direction) from the start position. So, the primary goal of this system is to optimize (maximize) the network coverage area of WSNs.

The authors of [RK04] also utilize the rearrangement of nodes but with the primary goal to reduce the cost of communication and mobility. The goal environment for this approach is a sensor MANET in command-and-control context. Each node in this network is supposed to be a robot (e.g. drones, UAVs) carrying one or more sensors. The authors distinguish two types of nodes: *tracking* and *scanning* nodes. The tracking nodes generate a large amount of a latency-critical traffic which can include real time information about the state of a tracked facility. The scanning nodes perform the sensing of the environment and generate a non-latency-critical traffic with a lower priority negligible in comparison to the traffic generated by the tracking nodes. The scanning nodes also relay the traffic between the tracking nodes and a data sink (a base station). In order to minimize the total energy consumption of the network and to increase the sensing area, the scanning nodes can change their own positions based only on the information from their local environment. In the scenario presented in the paper, the tracking nodes are positioned statically and the intermediate scanning (and relaying) nodes are rearranged to provide a path from the tracking nodes to the data sink, optimal considering transmission power.

Similar to the previous works, the authors of [GLM+04] propose to rearrange intermediate relaying nodes in order to improve communication performance between source and destination nodes. The authors discus a self-organizing mobility control algorithm. As

initial step, the algorithm discovers a routing path between a source and a destination node, which typically have statical positions, using a greedy geographical routing protocol. The nodes belonging to the routing path change their own positions to provide a more effective transmission with lower energy consumption and a higher bandwidth. Thus, the nodes building the path (in case of a single source and destination) tend to form a geographical line between the source and the destination. The authors also consider the case if there are more communication paths in the network. In this case, the resulting topology looks like a set of crossing segments defined by multiple sources and destination nodes.

A complex solution for the communication recovery in disaster scenarios is proposed by the authors of [AT04]. The authors rely on a previously developed Wide area Disaster information Network (WDN) which combines both wired and wireless networks in order to provide resident safety information system and bidirectional video communication between a disaster area and headquarters. The authors introduce the Wireless Recovery Protocol (WRP) aimed to provide a temporally solution for the recovery of WDN, if it is damaged by the disaster. A network management protocol together with GPS functions is used to determine the exact failure positions in WDN. If WDN detects a connectivity problem, the system uses the rearrangement of mobile terminal nodes to restore the failed links. As mentioned before, the wireless recovery protocol is a part of the disaster information network which should already exist before a disaster happens.

3.4.4 System Using Message Ferrying

A completely different solution to support the communications within a partitioned network is presented by the concept of the Delay Tolerant Networks (DTNs). Typically, the nodes belonging to one network partition generate messages and store them for some period of time. During this time, the network partition is waiting for a mobile node called message ferry, which picks up the stored messages and delivers them to other partitions. Obviously, such a kind of communication does not allow any real time data transmission or data streaming. An example of such a network is presented in [WL10], where the authors try to solve the problem of generating dynamic message ferry routes. The dynamic routing planning for the ferry nodes allows to minimize the delays for message delivery and to optimize the trajectories. Each group of nodes builds a cluster and elects a cluster head, which is responsible for communication with the ferry node. The cluster heads collect the messages to be transported by the ferry node and distribute the messages addressing any cluster member. As the message ferry node, any node type with controlled mobility, like ground robots or UAVs, can be used.

3.4.5 Comparison and Evaluation

This section provides comparison of the systems presented above. Altogether, ten systems are included in the comparison. For convenience reasons, their key futures are

aggregated in a table. The table consists of two parts (table 3.1 and table 3.2), each of them includes characteristics of five systems. The following key features are denoted in the tables:

- **IDEA** – solution idea exploited in the system;
- **CTRL** – characteristic of the controlling mechanism (centralized, distributed);
- **BHV** – system behavior pattern (reactive, proactive);
- **DET** – basis for partitioning detection, used in the system;
- **LOC** – mechanism of partition localization, if required by the system;
- **OVHD** – main sources of the overhead produced by the system;
- **NETW** – characteristics of the nodes and the network, which should be repaired;
- **NODE** – characteristics of the nodes, used for network connectivity restoration;
- **COST** – costs relative to used hardware (*low* – no additional hardware, *medium* – only a small number of additional devices, *high* – many additional devices, or expensive devices);
- **PROS** – advantages of the system;
- **CONS** – system disadvantages and drawbacks;
- **APP** – main system application fields, considered by design.

Note that the advantages and disadvantages of the particular systems are given with respect to the context of this work, and can be seen differently in different contexts.

Relating to this thesis, the most interesting system is presented in [CDB04]. The authors consider similar application scenarios, when isolated groups of nodes should be reconnected to provide communication possibilities between them. The groups members are not required to change their positions. Instead of that, additional nodes are used. Here, the role of additional nodes play UAVs, which are used for message forwarding between the groups. The system presented there is designed primarily for military usage. However, the system has some important drawbacks: first of all, the system seems to be centralized since the UAV placement positions should be calculated by a central entity which is also aware of the positions of the node groups. Additionally, the system supposes that the network nodes are equipped with two different network interfaces: one for communication between the devices and one for communication with the UAVs. It is impossible to use “ordinary” devices, like smart phones or laptops, within this system.

Table 3.1: Comparison of the network partitioning recovery systems. *Part I*

	[**DPS08**] *Gaps in WSN*	[**CDB04**] *Network of UAVs*	[**DPH05**] *Microrouting*	[**MCG09**] *LANdroid*	[**ASTU10**] *PADRA*
IDEA	additional nodes placement	additional nodes placement	supporting network of microrouters	supporting network of robots	nodes rearrangement
CTRL	centralized via BS	centralized	not needed – static deployment of multiple nodes	distributed	distributed
BHV	reactive	reactive	proactive	proactive	reactive
DET	via epoch number	via locations	not required	not required	via beacons
LOC	during movement of additional nodes	movement prediction	not required	not required	nodes neighborhood and location
OVHD	epoch numbers distribution and checking	location information updates, UAVs control	only routing protocol	robots coordination	beacons, high during initialization
NETW	static WSN with BS	groups of mobile nodes (MANET)	not restricted	not restricted	mainly statical MANET of WSN
NODE	additional movable nodes	UAVs	microrouters	robots	network participants
COST	medium	medium	high due to a high number of microrouters	high due to usage of multiple robots	low
PROS	simple, using additional nodes	using additional nodes, optimal number of additional nodes	simple, large coverage area	solid connectivity and coverage in urban areas	minimized movement of individual nodes
CONS	centralized	centralized, two interfaces by normal nodes	microrouters deployment, costs	costs	change of participant positions
APP	reconnection of static WSNs	military, connecting groups of ground nodes	military, large area sparse MANETs	urban areas, MANETs and WSNs	WSN with static sensors and movable actors

Table 3.2: Comparison of the network partitioning recovery systems. *Part II*

	[**WL09**] *Node allocation scheme*	[**RK04**] *Energy optimization*	[**GLM$^+$04**] *Mobility control primitive*	[**AT04**] *Wireless Recovery Protocol (WRP)*	[**WL10**] *Message ferrying*
IDEA	nodes rearrangement	nodes rearrangement	nodes rearrangement	nodes rearrangement	Delay Tolerant Network (DTN)
CTRL	distributed	distributed	distributed	distributed	distributed
BHV	proactive	proactive	proactive	reactive	reactive
DET	not required	not required	not required	resource information system	locations and clustering
LOC	no	no	no	resource information system and WRP	clustering areas are known
OVHD	beacons	beacons	beacons, movement control	control traffic of the whole disaster information system	cluster building
NETW	MANETs	tracking and scanning WSN nodes	WSNs	different types of nodes	groups of mobile nodes
NODE	network participants	network participants	network participants	network participants	message ferries
COST	low	low	low	high due to the disaster information system	medium
PROS	coverage maximization, low risk of partitioning	optimal positioning of scanning nodes, energy consumption	large areas, energy consumption	robust, heterogeneous networks	possible large distances between node groups
CONS	change of participant positions	change of participant positions	change of participant positions	change of participant positions; complex infrastructure	large delays, non-real-time communications, static cluster regions
APP	MANETs, monitoring, environment sensing	monitoring, environment sensing, facility tracking	large scale WSNs, long-term sensing and monitoring	disaster communication and recovery	disaster relief communications, wide area sensing

A big group of systems ([ASTU10, WL09, RK04, GLM+04, AT04]) considers network partitioning recovery or even prevention by rearranging the member nodes in one or another form. The systems also cover a wide spectrum of networks: from simple wireless sensor networks to MANETs and even completely heterogeneous networks ([AT04]). The main advantage of these systems is that the systems does not require any additional hardware being available. However, the solution methodology – rearrangement of the network participants – is not suitable for the usage scenarios considered in this work. As mentioned before in section 1.2, it is inadmissible to require, for example, a member of a rescue team to change his position with the only goal to restore the connectivity.

Another group of solutions exploits the idea of deployment of complete supporting networks, consisting of a number of simple relaying nodes ([DPH05]) or even of mobile robots ([MCG09]). Both solutions prevent the appearance of network partitions at all since the supporting networks are designed to provide a persistent coverage of the disaster area. However, they are very cost intensive due to a large number of additionally required devices.

The remaining two systems have relative narrow possible application scenarios. The system presented in [DPS08] also utilizes the concept of additional nodes placement, but it is centralized and designed primarily for statical wireless sensor networks controlled by a base station. Another idea is exploited in [ASTU10], where message ferries deliver messages between isolated network clusters. Such a kind of communication is suitable only for a restricted number of tasks where the delay between sending and reception of a message is not very important. Automatically, this communication pattern is not applicable for voice and video communication, and also for any communication requiring real-time data transmission, since the delays are often not constant.

3.5 Summary

Summarized, the common features of communication in disaster scenario were discussed in this chapter with respect to modern approaches developed for disaster management. It was shown, that communication systems play a vital role during rescue measures and disaster recovery. Without a properly working communication infrastructure, an efficient coordination of the rescue forces is not possible at all. It is obvious that in disaster scenarios it is necessary to use all available resources, including existing unimpaired communication infrastructure.

However, the existing communication infrastructure can be partially damaged, if not completely destroyed, by a disaster. A quick restoration of these damages can provide additional chances to victims during the disaster rescue operations. Many solution proposals are existing, aimed at recovering communication networks on-demand, some of them were discussed in this chapter. However, all of them have some drawbacks, which makes the usage of these solutions very restricted or even impossible with respect

to scenarios addressed in this work. The rest of this thesis is aimed at describing a proposed alternative solution, which tries to eliminate these drawbacks and to take the best from the discussed systems.

4 System Concept and Architecture

In this chapter, the main requirements for the developed system are presented, and assumptions and restrictions are discussed. Also, two solution methodologies are described here: for the case, if global knowledge about the network topology and node positions within the network is available in some form, and for the case when such knowledge is not available. Then, the interface for the developed system is defined, and the integration of the system into an after disaster recovery system is shown. In the remainder of this chapter, two use cases for the system are discussed, too.

4.1 System Description and Goals

The main purpose of the described system is the recovery of a MANET, whose functionality can be unavailable due to some reasons. First of all, the network can be partitioned, when some nodes or group of the nodes become isolated from others. Such a scenario can be assumed easily if different groups of one rescue team during a disaster recovery veer away from each other, and communication between them is not physically possible any more.

Furthermore, the network functionality can be broken if a base station fails in a heterogeneous or infrastructure-based network due to some damages during a disaster. Also, such problems like a failure of an important node, of a server, of a service or of a link can lead to similar problems.

4.1.1 Prerequirements and Main Assumptions

Here, some prerequirements and assumptions for the developed system are defined. First of all, it was decided that the system should have a reactive behavior. It means the system becomes active only after some failure or problem was detected. The reason for this decision is the goal to minimize the communication overhead produced by the system and to save the resources of the network (especially, in terms of energy).

The second assumption is that there can be no central controlling element available in the network, which is typical for MANETs. Therefore, the algorithms used by the system should be fully distributed with self-organization functions. In such a situation,

each node of the network is an element of the self-organizing system and as such an element, it uses only the locally available information about its environment, such as the information about its neighbor nodes.

Of course, if some global knowledge about the network topology is available in the network, this information should be also taken into account, because this can affect positively the accuracy of results – e.g. the accuracy of the computed position for the placement of additional nodes for network recovery. This situation can be possible, if, for example, a central computer in a head-quarter of a rescue team can be used as a central controlling element for the network. In this case, it is also possible to apply some centralized algorithms to calculate the placement positions more precisely.

Since MANETs consist of mobile movable nodes, the algorithms used in the system should also be able to work with a dynamically changing network topology. According to the main usage scenario of the system, communication in a disaster scenario, the maximal velocity of the nodes is assumed to be about 20 m/s. Note that this speed value corresponds to the value which exceeds the average speed of human being running on short distances almost twice.

Within the scope of this work, it is assumed that the network to be repaired is an IP-based network. However, the proposed concept can be extended for others types of networks e.g. also for infrastructure-based networks like cellular mobile phone networks.

The proposed system concept assumes also that some additional freely movable mobile nodes which can be used for network recovery are available in the system. These nodes can be drones, Unmanned Aerial Vehicle (UAVs) or any other devices meeting the two following requirements:

1. the device should be compatible with the network to be repaired and can provide the missing functionality;

2. the device can be placed into a specific position requested by the system.

The assumption of the presence of some additional nodes which can be used for network partitioning recovery is based on the fact that some additional equipment is typically available for disaster recovery operations.

4.1.2 Functional Requirements

In this section, the functional requirements for the developed system and for the nodes which are members of the network to be repaired are described.

First of all, the system should provide the ability to detect a problem appeared during data transfer between the nodes. For this problem detection, the system should not react to ordinary link failures which are caused by a changing network topology, when a

new route to the destination can still be established. Thereby, a trigger event activating the system should be generated if and only if it was impossible to repair the route.

The next task for the system is to determine where a gap in the connectivity has emerged to calculate the placement positions for additional nodes. At the same time, it is necessary to recognize the situations, when the route cannot be reestablished due to the failure of the destination node. In this case, the placement of additional nodes is not meaningful.

After the goal position for the nodes placement is calculated, the system should initiate a request to place the nodes addressing a mission planning subsystem (for example, a server which controls the flights of UAVs). In the simplest case, the system should just send the request directly to a movable node, which should then advance to the specified position.

Finally, if the network becomes split into two partitions only, it can lead to many communication errors occurring simultaneously. In this case, the system should be able to combine similar problems into one aggregated problem to be solved.

Another group of functional requirements is related to the nodes of the network to be repaired. These requirements should be fulfilled so that the partitioning recovery system can work properly.

To make it possible for the nodes to determine their own position, it is necessary that a location system is available within the network. In the simplest case, the location system can be provided by equipping all the network nodes with GPS devices. This requirement is feasible for most of the modern devices since they have an on-board GPS device already built in.

The nodes of the network to be repaired should also keep track of their immediate neighbors, that is often already done by routing protocols which are used in MANETs.

Additionally, the nodes should track their communication partners – the end-nodes of an IP-based communication. For this purpose, the information about the geographical location of the partner nodes can be distributed within an additional options field in the header of IP packets transmitted by the nodes. How it can be done exactly will be discussed later in section 5.5 for both IPv4 and IPv6.

Of course, if some kind of global knowledge about the network topology is available in the network, it is not necessary to transport the geographical coordinates in IP headers since it is already a task of another more sophisticated location system to keep track on the positions of all nodes belonging to the network. In this case, the location information is typically available for all network participants.

4.1.3 Non-Functional Requirements

Along the functional requirement described above, there are also non-functional requirements for the system which should be taken into account during the development of the system, too.

One of the aspects is the goal to reduce possible communication overhead during the system work. Ideally, without problems being detected the system should not produce any overhead at all or keep it as low as possible. Also in active state, additionally generated traffic should not disturb the network.

The second aspect is that the system should be independent of the type of routing mechanism used in the network. This can reduce the dependencies between different communication planes for the partitioning recovery system and makes it more flexible regarding underlaying network technologies.

4.1.4 Communication Planes for Partitioning Recovery

To make the proposed concept of the partitioning recovery system more flexible, all communications in the network are divided into two classes logically: repair plane communications and usage plane communications (see fig. 4.1).

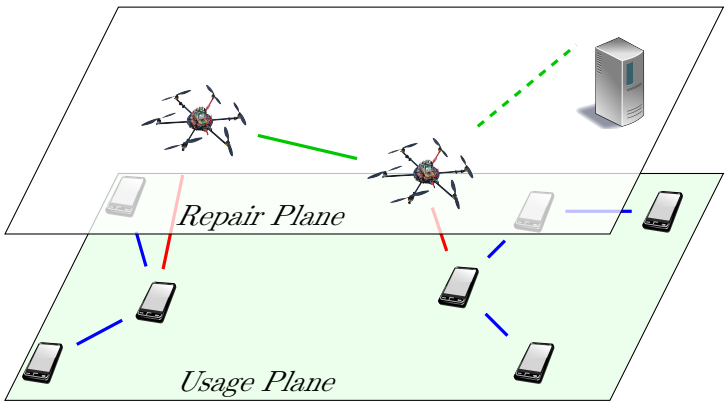


Figure 4.1: Recovery and usage communication planes

Repair Plane All communication needed for problem detection, position calculation, and mission requesting belong to repair plane communication. An exception here is communication during a routing discovery process since the routing is a part of the network to be repaired and is independent of the partitioning recovery system. In figure 4.1, the following components belong to the repair plane network: two UAVs which provide the functionality of relaying node, and an optional server which coordinate the UAVs if a centralized approach is used.

Usage Plane All the communications transmitting user data and also control traffic supporting directly the functioning of the underlaying network, which is not related to the network partitioning recovery system, belong to the usage plane communication. In figure 4.1, the usage plane network is presented by two isolated sets of mobile nodes (smart phones).

Similarly to the communication planes, the network can also be divided logically, or even physically into two parts: user network, or network to be repaired and rescue network. The user network is the network providing the functionality to fulfill the users requirements and to transport their data. User services also belong to the user network. The rescue network is responsible for the data transmission needed for the partitioning recovery system and for the communication between a control server and the additional mobile nodes if the control server is used in the network.

For the actual work, both networks are based on WiFi technology, however the additional nodes are equipped with two interfaces to separate the networks. Since both network are WiFi networks, a single WiFi interface is also sufficient for this work. In this case, the networks can be separated by using different addresses or subnetworks.

Theoretically, both networks can be based on different wireless networking technologies. For example, the user network is a UMTS cellular network, and the rescue network is based on WiFi communications. Of course in this case, the additional nodes to be placed for the network recovery should be able to communicate with both types of the networks: the additional nodes are controlled over the WiFi interface and provide simultaneously the base station functionality for the user network.

4.2 Availability of Global Knowledge

The main application scenario for the developed system is the recovery of communications in disaster scenarios. In such scenarios, a number of additional technical resources is typically involved into clearing disaster consequences. As a rule, a dispatching center or a headquarter of the rescue forces is organized in the field for after disaster recovery. The additional technical tools make it possible to localize victims of the disaster using mobile devices which the victims probably have.

In this case, the headquarter has the information about the devices which the communication may be necessary for. Hereby, the coordinates for the placement of additional nodes can be calculated more easily and more precisely using the available localization information. It is reasonable here to perform the calculation in a centralized way on a central computer of the headquarter, and the mission requests for the placement of additional nodes should be generated also by this central computer and not by the nodes itself.

Since the global knowledge can be available or not, there are two possible methods to solve the task of the network partitioning recovery: a distributed and a centralized approach.

4.3 Solution Methodology

The general scheme of the partitioning recovery system architecture is shown in fig. 4.2. The main task of the system is the partitioning recovery (*Partitioning Recovery* task).

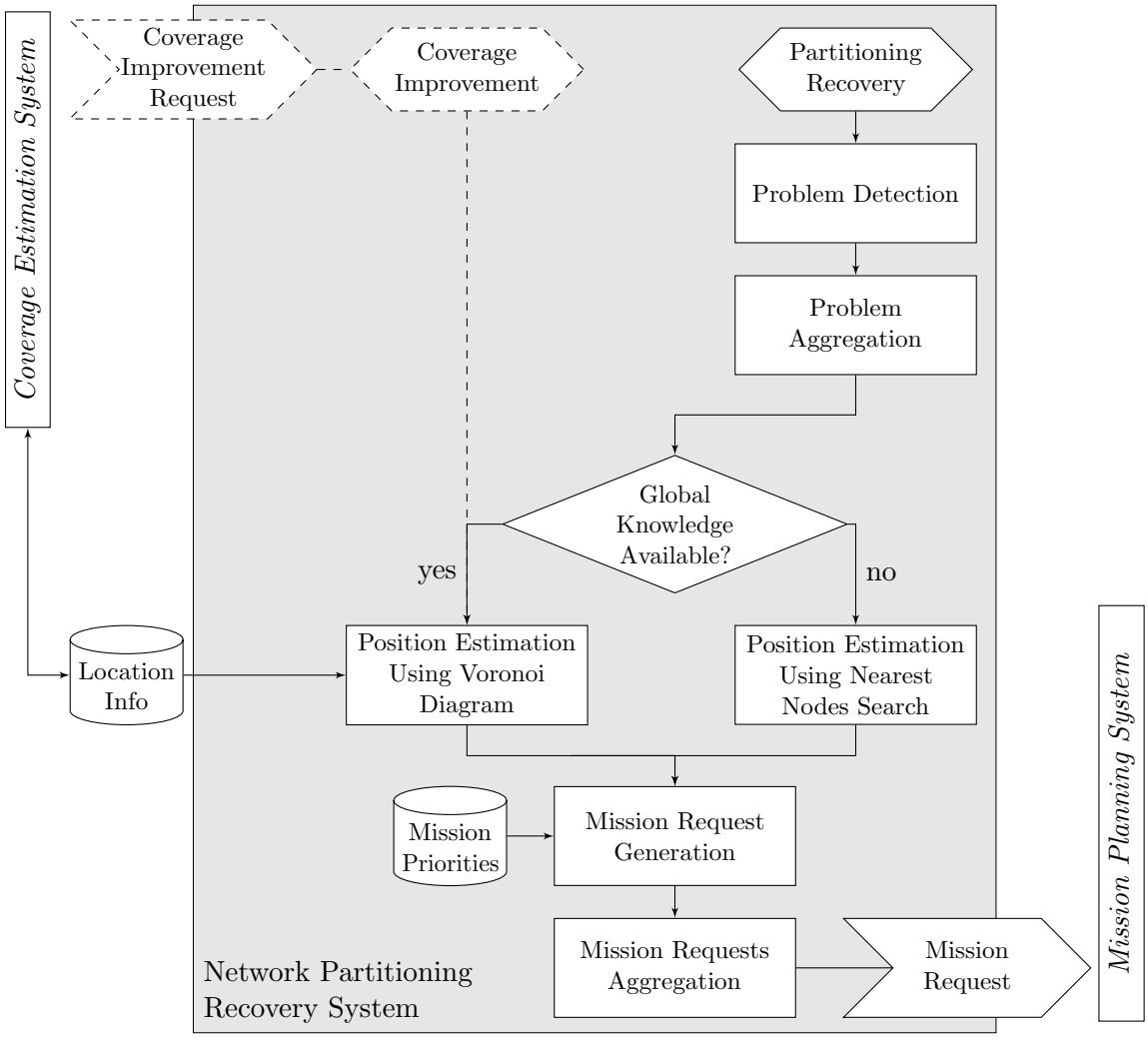


Figure 4.2: System architecture and system borders

The first step in this direction is to detect the problem existence (*Problem Detection*). As mentioned before, even separation into only two isolated partitions can cause a number of problems which are detected by the system. Therefore, as the next step, the system performs an aggregation of the detected problems (*Problem Aggregation*).

Then, the system has to decide how the placement positions for the additional nodes will be calculated depending on the availability of global knowledge about the network topology. If global knowledge is available (*Location Info*), the system calculates the placement positions using the centralized approach (*Position Estimation Using Voronoi Diagram*, see section 5.8 of the following chapter). If there is no global knowledge available, the system utilizes a distributed approach (*Position Estimation Using Nearest Nodes Search*, see section 5.6 of the following chapter).

Based on the calculated placement positions, the system generates a mission request for the placement of additional nodes (*Mission Request Generation*) and assigns some priority value (*Mission Priorities*) for it.

As in case of the distributed mission calculation several mission requests can be produced in different parts of the network, these mission requests should also be aggregated if possible. Thereafter, the system sends out an aggregated mission request for the placement of additional nodes to the mission planning subsystem which controls mission execution.

Finally, the developed system can also be utilized with some minor extensions for coverage improvement of a wireless network. This part of system is shown in fig. 4.2 with dashed lines (task *Coverage Improvement*). An input interface (*Coverage Improvement Request*) belongs to this part of the system. The coverage improvement request triggers the system to perform the placement calculation for coverage improvement. It is supposed that global knowledge about the network topology is available for the system. Based on this knowledge, the system calculates the placement position for additional nodes and generates the corresponding mission request.

4.4 System Interfaces and Integration into the Disaster Communication System

Now it is time to define system borders for the partitioning recovery system. The borders separate the system from other network components and subsystems. In figure 4.2 the system borders are shown as gray background rectangle, and all the parts within the rectangle belong to the system.

The system should be able to communicate with other network subsystems and a corresponding communication entity should be integrated into the network protocol stack of each node belonging to the system. Thus, two principle communication ways between the network partitioning recovery system and the environment are possible. Firstly as a part of the protocol stack of a node, the system interacts with ingoing, outgoing and transit messages of the node, and can intercept and change them if necessary (vertical communication). Secondly, the system instances in different

nodes are interacting with each other and with other network subsystems by sending communication packets over the network (horizontal communication).

In figure 4.3, the system integration into the network protocol stack is shown. The stack model is based on the Internet Protocol Suite [Bra89] model. The main part of the system entity *SYS_MAIN* is placed between the Internet (IP) and the Transport (TCP/UDP) layers. Such a position provides access to all IP packets traversing the protocol stack of the node. For example, this makes it possible to extend the headers of the outgoing IP packets with additional data containing the geographical position of the node. Furthermore, it functions transparently for the upper layers and applications because the system detaches this data before the packet goes further up in the protocol stack.

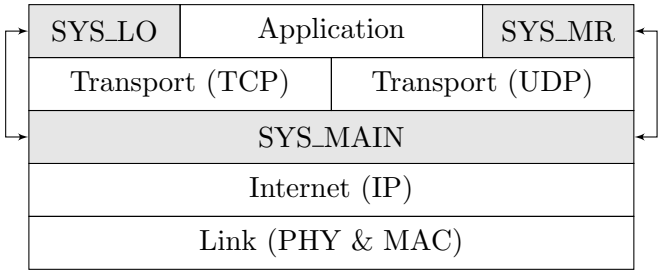


Figure 4.3: Place of the partitioning recovery system in Internet protocol suite

On this level, it is also possible to monitor traversing ICMP error messages, which are generated by a routing entity when it has not found any route to the specified node. In cooperation with the communication partner tracking mechanism provided by the system, a communication problem can be detected by the node, that can indicate network partitioning.

There are also two additional blocks belonging to the system which are placed in the application layer. Block *SYS_LO* provides the communication interface to the network participants which offer the access to the global knowledge about the network topology, if it is available within the network. As a data storage for the global locating information, both the centralized, and the distributed data base can be used. In both cases, the interface to the global information is the same and will be discussed later in section 5.7 in more detail. Thus, for the system it does not matter how the global topology information is stored.

Block *SYS_MR* is responsible for the generation and the aggregation of mission requests, which are sent out to the mission planning system controlling the placement of additional nodes. During the placement position calculation in case no global knowledge is available, the network load increases significantly, and the mission requests should be aggregated. Because of this, it is more reasonable to use an unacknowledged data transmission based of UDP. It provides the ability to reduce the communication overhead caused by retransmission of packets with lost acknowledgements as it would be a case during the

usage of a TCP connection. At the same time, the loss of some mission request packets is not critical for the system because several mission requests will be generated with a high probability anyway.

4.5 Unit Graph Model

For modeling the topology to develop and simulate the algorithms, a unit graph model was applied, which is widely used in researches of networking algorithms (e.g. [KK00, BMSU99]). This model is based on graph theory, and the network topology can be presented here as an undirected graph on plane or in space. The network nodes are the vertexes of the graph, and the vertex's position on plane (or in space) is defined by the geographical position of the corresponding network node. Any two graph vertexes are connected with an edge if and only if the Euclidean distance between them does not exceed some predefined value R (communication radius) which corresponds to the communication range of the node. In the unit graph model, the value of R is assumed equal for all nodes in the network. The unit graph model is illustrated in figure 4.4: the distance between nodes A and B is lower than the communication radius R so the nodes A and B are connected in the graph, and the distance between nodes A and C exeeds R and the nodes are not connected.

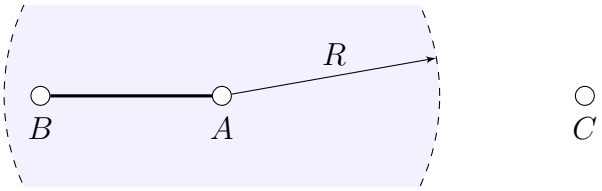


Figure 4.4: Unit graph model

Such a model provides a rough view on the network topology and does not take into account different physical aspects of a wireless communication. Firstly, a fixed value of communication radius R implies that the radio signal of a node propagates uniformly in all directions and the coverage zone has a circular shape in this case, which might not reflect the reality in some cases. Secondly, the unit graph model does not consider any obstacles which can block the connection between nodes. Thirdly, all links are considered as symmetric (bidirectional).

In spite of the restrictions listed above, this model is effective, and first of all made for simulations. It allows to avoid calculation of complex physical models of radio signal propagation. For distributed algorithms using the unit graph model, a fixed value of R is also not critical because only the existence or the absence of a link is considered. The same thing is valid for a situation with obstacles, too.

Asymmetric links can cause some problems for a transmission without acknowledgement such as in beacon messaging. In this case, the node can assume accidentally that there

is a bidirectional link with the neighbor which has transmitted a beacon message when the link is only unidirectional. This problem can be avoided simply by including the information about already known neighbor nodes into each outgoing beacon message. Thereby, the node which receives a beacon assumes a bidirectional link only if it sees itself in the neighbor list of the received beacon message.

4.6 Use Cases

At the end of this chapter, two main use cases for the developed system are explicated.

4.6.1 Partitioning Recovery

The first use case is partitioning detection and recovery in MANETs. The matter of this use case is not only the network recovery after a disaster, but also the partitioning recovery for regular MANETs. The only restriction here is the availability of additional mobile nodes, which can move to a specific geographical position by request of the system. These nodes can also be regular nodes, if they can change their position by request if it is allowed by the usage model of the network. For instance, it might not be possible to request one of the rescue team members to go to a specific position just to restore network connectivity since it will collide with the main tasks of the rescuer.

4.6.2 Coverage Improvement

As mentioned previously, the system can be also used for coverage improvement of a wireless network. With a mobile base station, for example, a fixed base station destroyed during a disaster can be replaced. However, this use case is more related to network recovery after the disaster because it requires that the network coverage is estimated in advance. It makes sense within a disaster recovery scenario since a preliminary reconnaissance of the situation and a damage estimation are performed typically before the recovery works start.

4.7 Summary

Summarized, the system architecture, the integration and the interaction with other network subsystems was described in this chapter. In the following chapter, details of the technical realization of all architectural elements (blocks) presented here will be considered in more detail.

5 Architecture Blocks

This chapter deals with the description of the particular architecture blocks of the proposed system for network partitioning recovery. Here, technical aspects of the realization of each component (architecture block) of the system is given, too.

For the better understanding the technical background of the components described in the following sections, the chapter starts with a dedicated section which introduces the software framework used for the implementation of these components.

5.1 Click Modular Router

This section provides some background information about the technics used for the implementation of the particular architecture blocks. The Click Modular Router [KMC+00] is used in this work as the main software platform for implementation and configuration of the network nodes.

5.1.1 Common Description

Click is an open source, flexible, modular software architecture for creating routers [KMC+00]. Click makes it possible to build a router in a Linux-based operating system from a number of interacting components which are called elements. These elements are composed into a directed graph according the packet data flow of the router.

The elements are the building blocks which each Click router is constructed from. Each Click element provides its own particular function: for example, one element can only set the source or destination IP address of an IP packet, another can be used to calculate the checksum for this packet. Using the particular functions of the Click elements, it is possible to build very flexible configurations of routers, which can be easily extended or modified. Most elements accept configuration parameters via configuration strings. A configuration string can influence the particular behavior of the element or provide the element with some data relevant for processing the packets.

Most of the elements have input and output ports which the packets are directed through. However, there are some elements having only input, or only output, or even

no ports. Depending on the behavior pattern, Click supports two kinds of connections: push and pull. The most common connection is the push connection. In this case, a source element passes the packets downstream towards their destination. By the pull connection, the packet transmission is initiated by the destination element. Click allows only the connections between two push or two pull ports. To avoid this limitation and to make interconnection of elements more flexible, a special type of ports is defined in Click – the agnostic ports. An agnostic port connected to a push port behaves as a push port, and connected to a pull port as a pull port.

To interact with the router and to adjust the behavior of some elements of the router from the outside, Click offers the mechanism of handles for the elements. There are read only, write only, and read and write handles. The handles are accessible via Linux sockets by names. A read handle can be utilized, for instance, to obtain some statistical information, like packets count, from the router, or to check the content of a routing table. Through a write handle, particular router elements can be configured on-line without stopping the work of the router. So, for example, a new route entry can be appended to a routing table, or an existing route can be deactivated.

There is also a way for direct interaction between different elements, not using packets transmission. An element can provide an arbitrary interface to the others which is called method interface. For this purpose, any arbitrary public method can be defined within the C++ class of the element so that the other elements can get access to it. The method interface is especially suitable for elements maintaining data like routing tables or addresses of some hosts. Such elements do not process any packets directly, but are used for packet processing by many other elements. However, an overuse of the method interface can significantly increase the dependency between the elements, which can negatively affect the flexibility of the software router configuration.

New elements can be simply added to Click by subclassing of the *Element* class provided by the Click API. Each element class should implement at least one method which returns the name of the element. Obviously, the methods returning port count, processing type (push, pull or agnostic) for the corresponding ports are implemented along with methods for configuration, initialization and push processing.

In Click, two working modes are available: as a user space application and as a Linux kernel module. The user space mode is useful for router configuration debugging opposite to the kernel module, whose malformed configuration can lead to a crash of the whole system. The kernel module provides higher performance in contrast to the user space mode. However, a significant increase of performance can become visible only for highly loaded routers with high throughput and big traffic volumes. Therefore, the user space mode is chosen for the implementations and tests of the system to keep it simpler and to avoid the problems and the restrictions caused by executing the program in kernel space.

Click provides a set of API to make possible to implement some functionalities available also for kernel space mode which are not supported directly by the Linux kernel API

since the standard library of C++ is not available in kernel space. First of all the classes implementing the commonly used complex structures like arrays, strings or hash tables belong to this API. Click provides also a mechanism of high resolution timers which are widely used in protocol implementations. The availability of this API in both the user space and the kernel space modes makes the usage of the API preferable for implementation since the elements relying only on this API can be accessible in both modes.

A router graph specification is written as a text file using a specific language. This language also allows to build groups of elements and to use configurations consisting of multiple files. An example of a simple router graph is shown in fig. 5.1. This graph consists only of three elements: *FromDevice*, *Print* and *Discard*. The *FromDevice* element provides an interface to an existing network interface device registered in Linux. The packets arriving from this device are pushed to the next element *Print* which prints out into standard output a given number of bytes (24 by default) of the packet content with some given text label. The last element *Discard* just drops all arriving packets.

Figure 5.1: Example of a simple Click router graph

The router graph shown in fig. 5.1 is a good way to graphically present the entire structure of a software router and the interconnections between the router elements. However, it does not show any configuration parameters given to the elements in the router configuration script. The router configuration script corresponding to the graph shown in fig. 5.1 is shown in the listing of fig. 5.2. From the source text of the router description script it becomes clear, that the router accept the packets from the network interface `eth0` and the output for packet content is labeled with text "Ok".

```
1 FromDevice(eth0)
2   -> Print(Ok)
3   -> Discard;
```

Figure 5.2: Example of a simple Click router script

5.1.2 AODV in Click

As mentioned in the previous chapter, the AODV routing was chosen for this work as the basic routing mechanism. Its Click implementation was proposed by Bart Braem in [B. 05]. It provides the AODV Click elements themselves and some Click configuration scripts for a Linux system and also for the NS-3 simulator. The software

is an open source project published by the Performance Analysis of Telecommunication Systems (PATS) research group at the University of Antwerp [Bra09].

The AODV elements provide a simple realization of RFC 3561, without implementation of optional features. The AODV Click elements with a short description are listed in the following table 5.1.

Table 5.1: List of available AODV Click elements

Element [Bra09]	**Description**
AODVDestination-Classifier	classifies `RREP` AODV packets on destination into the packets generated by the host itself, the packets which a route is already known for, and the packets which a route is unknown for yet.
AODVGenerateRERR	generates AODV route error (`RERR`) packets according to section 6.11 of RFC.
AODVGenerateRREP	generates AODV route reply (`RREP`) packets according to section 6.6 of RFC.
AODVGenerateRREQ	generates AODV route request (`RREQ`) packets according to section 6.3 of RFC.
AODVHelloGenerator	periodically generates AODV `HELLO` packets according to section 6.9 of RFC.
AODVKnownClassifier	classifies route request packets according to the requesting destination IP address for the host itself and for the others.
*AODVLinkNeighbours-*Discovery	links *AODVNeighbours* with *AODVWaitingForDiscovery*, so discovery knows when a new route is added which might release waiting packets. It does not process any packets directly.
AODVLookUpRoute	determinates whether the route for the incoming packet is known or not. If the route is known it moves the packet to first port otherwise to second or to third port depending on the packet source (local generated or transit).
AODVNeighbours	maintains the routing table and provide access to it via a method interface to the others. It does not process any packets directly.
AODVSetRREPHeaders	sets an IP header for the outgoing route reply packets.

Table 5.1: List of available AODV Click elements (continued)

Element [Bra09]	Description
AODVTrackNeighbours	processes all IP packets and updates the routing table if necessary.
AODVUpdateNeighbours	processes `HELLO` packets and updates the information about the neighbors.
AODVWaitingForDiscovery	collects and holds the packets a route is being discovered for.

However, some problems were faced with the AODV elements: sometimes the program failed with assertions of bad destination IP addresses for some packets in queue to be routed. According to the packet trace log collected by Wireshark [C+12], some routing loops were built in such cases. To fix this issue, it was forbidden to the nodes to answer with `RREP` on `RERR` messages, if these messages arrive from the next hop node for the corresponding route.

For the convenience of deployment, the AODV elements were organized in a separate package and the author was provided with the corresponding patches. To use the AODV elements in Click configuration script, the package has to be included into the configuration file using the following string:

```
require(package "AODV");
```

Some other implementation details for AODV in Click will be given, whenever they are relevant for some architecture blocks.

5.2 Problem Detection

The first element, which brings the system to the active state, is a problem detector. In this section, it will be described, how the problem detector works in the system and which requirements it should fulfill. The main goal of this problem detector is to detect that an existing communication is broken and cannot be restored using regular routing mechanisms present in the network. However, it is not aimed to detect whether the network was partitioned or it was just a single node failure. This decision will be made by other components of the system.

5.2.1 Existing Approaches

A number of methods for connection failure detection are presented in literature both for MANETs and for Wireless Sensor Networks (WSN). The whole multitude of methods

can be divided into two classes: pre-detection (or prediction) systems and post-detection (or simply detection) systems.

5.2.1.1 Prediction Systems

Pre-detection systems apply methods of graph theory to detect where in the network a partitioning might take place. The authors of [SLS06] propose to detect a critical node within the network whose failure can lead to network partitioning. They present the DMCC (Detection algorithm based on Midpoint Coverage Circle) algorithm which is aimed to find a global critical node for the network. The authors define three types of nodes: global critical nodes, local critical nodes, and ordinary nodes. The failure of a global critical node leads to network partitioning. At the same time, the failure of a local critical node builds a coverage gap, but the network is still connected. However, for a network with mobile nodes, the DMCC procedure should be repeated continuously to consider changes in network topology.

In another project [HCS03], the authors utilize the idea of disjoint path sets. The fewer alternative paths between nodes exist, the higher the probability for network partitioning between these nodes is. The proposed system does not utilize any positioning system. The system tries to find as many disjoint patches between nodes as possible using a controlled flooding mechanism. The disjoint path calculation procedure is led by two nodes which are server and client. Depending on the algorithm parameters, it may not cover the whole network but only a part, the server and the client belong to. Thus some election mechanism is needed to assign the roles of client and server to the nodes, and the partitioning can be predicted only between the client and the server. Presence of more clients and servers leads to producing a significant communication overhead, and to high network load especially in a mobile environment.

Another partitioning detection approach is presented in [RC08]. There, a Connectivity Index (CI) is proposed to be used for partitioning prediction. However, the authors propose only a mathematical model and this approach seems to be centralized, that leads to a single point of failure.

5.2.1.2 Detection Systems

The Post-detection systems are monitoring the whole network continuously. Authors of [CSAB08] proposed a mechanism for failure, disconnection and partition detection in a mobile environment. They consider the network as a distributed system of processes exchanging messages with each other. The authors introduce a system architecture consisting of several building blocks: a Heartbeat Failure Detector (HBFD), a Vector-Based Disconnection Detector (VBDD), and an Eventually Perfect Partition Detector (EPPD). In terms of interacting processes, the HBFD provides each process with information about the neighbor processes and the processes, which can be reached over

the neighbors. The VBDD provides the processes with a vector containing information about connection/disconnection events of a process. It is assumed that the process does not crash, and notifies the others about its own connection/disconnection. The EPPD exploits the information provided by HBFD and VBDD to inform the process about suspected processes which are not in the partition of the current process. The EPPD also monitors disconnection events from the VBDD.

Authors of [RWS08] presented two algorithms aimed to detect the network partitioning. One algorithm is centralized and other is distributed. In case of the centralized approach, a server is periodically exchanging messages with some active nodes in order to test the connectivity with them. In this case, the server maintains a list of active nodes. However, if there are no active nodes within a partition, the partitioning will not be detected. In the distributed approach, the active nodes are exchanging the messages between each other. So, some nodes get additional roles from the network to support the partitioning detection system. Additionally, an election procedure is needed to decide which node becomes active, and the election procedure leads to a broadcast storm shortly after startup.

In [SST05], the authors proposed a solution of a problem similar to the network partitioning problem for WSNs. They described a method for detection of ε-cuts in sensor networks. For a network containing n nodes, an ε-cut is a linear separation of εn nodes ($0 < \varepsilon < 1$) from a distinguished node, the base station. The authors showed that an ε-cut can be detected using only a small number of sentinel nodes, the nodes which are distributed over the network in a special way. However, the proposed approach considers a centralized scenario using the base station as a central controlling element, and does not take into account the possibility of the nodes' movement.

5.2.1.3 ELFN

Another family of mechanisms for link breakage detection is Explicit Link Failure Notification (ELFN) [HV02] proposed for TCP and mentioned previously in section 2.3.6. Many researches demonstrate that the performance of a TCP connection degrades significantly with introduction of nodes' mobility due to high rate of packet loss for data and acknowledgment. A lost acknowledgement is interpreted by the TCP congestion control mechanism as an indicator of congestion and the transmission rate is slowed down. The proposed ELFN extension makes it possible for a TCP sender to distinguish a link failure from traffic congestion in that way, that an intermediate node detecting the link breakage explicitly notifies the TCP sender about the failure. In this case, the TCP sender is able to freeze its own retransmission timeout and window size by switching into a "freeze" state.

However, the ELFN approach is not reasonable for the implementation of the problem detector for the proposed network partitioning recovery system, since firstly the ELFN is designed only for the TCP connection, and secondly it does not make it possible

to detect whether an alternative route for the destination can be established or not. For the proposed system, this means a false positive response of the problem detector. Certainly, the proposed system does not interfere with the ELFN mechanism eventually used directly according to its intended purpose of performance optimization of a TCP connection within a mobile environment.

5.2.2 Goals and Requirements

The main goal of the problem detection component is notification of the network partitioning recovery system about a communication problem (break of an existing link) if the link cannot be reestablished over an existing or a new route. The generated notification triggers the start of the network recovery process.

As input, the problem detector uses locally collected information about the existing node's connections and observes the local communication situation by intercepting error messages and notifications processed by the host. The error notification within a node is basically provided by two mechanisms: by messaging over the Internet Control Message Protocol (ICMP), and by routing error notification (RERR) which is used by many routing protocols. Some implementations of routing protocols (for example, AODV for Click) generate also an ICMP message to the host if they cannot find a route to some destination.

Analysis of error messages and notifications and combining them with state information for existing connections make it possible to detect a probable network partitioning by the node. A communication tracking mechanism described below in section 5.2.3 is responsible for collecting the connections state information.

Output of the problem detector is an error notification containing the address of the unreachable destination. Depending on the technical realization of the system, the error notification can be implemented in several ways. For the implementation, proposed in this work it has form of a UDP datagram for centralized scenario and form of an API call within the node for distributed scenario.

As opposite to existing partitioning detectors from section 5.2.1, the proposed detector does not require any additional communication to be able to detect the problem. Only a small (about 12 bytes) piece of information should be piggybacked to some IP-packets by the communication tracking mechanism.

However, the proposed problem detector has some drawbacks. Firstly, the partitioning will not be detected by the node if it does not have any existing connections with nodes from the separated partition. Secondly, detection is possible only on side of a client since server side is not informed whether the client just has no more data to send, or it needs some more time for data processing, or it was disconnected during a partitioning.

In the next section, a pragmatic realization of the problem detector used in this work will be presented in detail.

5.2.3 Communication Partner Tracking

The communication partner tracker is a vital part of the problem detector. This mechanism supports the problem detector by providing the information about active connections of the current node. A communication partner of the node is a node in the network, which the current node has an active communication with. In other words, the communication partner is the end point of an existing IP communication.

The information about communication partners is stored in a table containing the following fields:

- **IP Address** – IP address of the communication partner node;
- **Latitude** and **Longitude** – last known geographical position of the communication partner identified by its IP address, this information is obtained from the extended IP header (s. section 5.5);
- **Timestamp** – time stamp of the last update of the record;
- **Status** – current status of the connection with the communication partner.

An entry into the communication partner tracking table is firstly done when the first packet to or from an unknown node traverses the protocol stack of the current node. The state assignment is implemented according the scheme illustrated in fig 5.3. In the figure, the current node is depicted as *me* and the communication partner node as x.

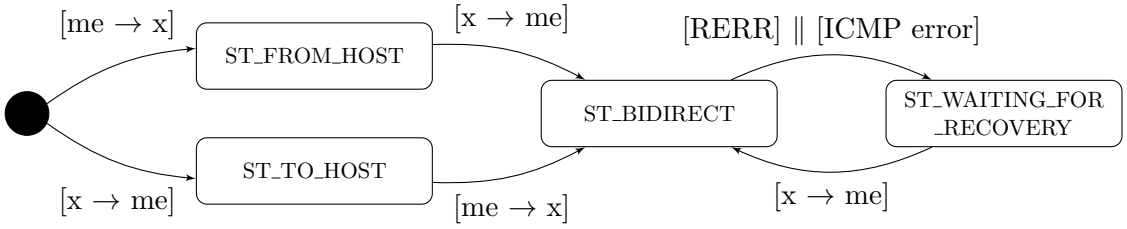


Figure 5.3: Connection states from communication partner tracking table

Depending on the destination of the packet, the state *ST_FROM_HOST* or *ST_TO-_HOST* will be assigned to the new entry. On the arrival of a packet to or from the communication partner respectively, the connection state will be changed to the *ST_BIDIRECT* state. Since the state of the connection is set to *ST_BIDIRECT*, an RERR or an ICMP error message corresponding to the destination will lead to immediately switching of the state to the *ST_WAITING_FOR_RECOVERY*, and the triggering event for the partitioning recovery will be released.

To keep the communication partner tracking table free from outdated entries, the table is regularly cleaned from old records with some given clean up interval. The only exception here are the entries marked with the state *ST_WAITING_FOR_RECOVERY*. So, the communication partner tracking table does not grow with time and the size of the table is determined by the number of existing communication partners.

The implementation of the problem detector is divided into two parts presented by two Click elements: the *CPTracker* and the *ProblemDetector*. The elements and their integration into the Click AODV router graph are illustrated in fig. 5.4.

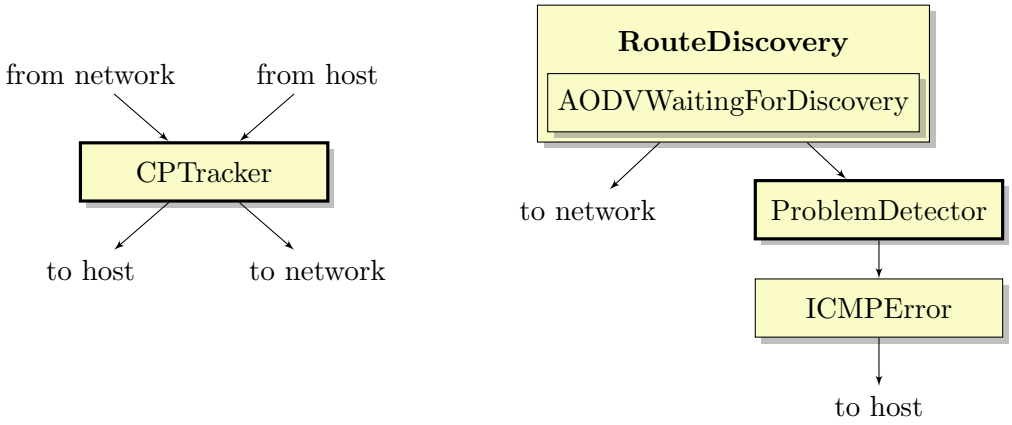


Figure 5.4: Problem detector within the AODV Click router graph

The first part – the *CPTracker* is deployed into the part of the Click router which is responsible for the forwarding of data packets from and to the host. The *CPTracker* element maintains the communication partner tracking table and provides the access to this table for other Click elements involved into the system for network partitioning recovery. The *CPTracker* has two input and two output ports corresponding to the two data stream directions: from the network to the host and from the host to the network.

The second part – the *ProblemDetector* element is integrated into the routing part of the Click router. The data packets which should be sent out are collected by the AODV Click element *AODVWaitingForDiscovery* within the *RouteDiscovery* mechanism. Normally, a waiting data packet is sent immediately to the network since the corresponding route was established. If the route cannot be established, the *RouteDiscovery* mechanism notifies the host with the ICMP error message using the *ICMPError* element that the destination is unreachable. The *ProblemDetector* reacts to this event and, if the state for the corresponding destination from the communication partner tracking table is set to *ST_BIDIRECT*, generates an error notification event which initiates the recovery procedure.

Both elements the *CPTracker* and the *ProblemDetector* do not change the transit packets but use the information contained in the packets to keep track of the existing communications and to indicate the problems caused by the possible network partitioning.

5.3 Beaconing

As a member of a self-organizing system, each node of the network needs to be aware of its local environment. In terms of communication networks, the local environment of a node is determined by its neighbors. The neighbors discovery mechanism is a vital part of many algorithms applied to wireless networks. First of all, such mechanisms are used by different routing protocols. The main principle of these mechanisms is very simple: the nodes periodically broadcast a message containing information identifying the node. In simplest case this information only contains a kind of identifier of the node like its IP address, or another address or name being used in the network. Depending on the usage, these messages can also include some additional information needed for the algorithm or for the application.

Such messages are called differently for different approaches utilizing them: `HELLO` messages or `BEACON` messages. In this work, a similar mechanism is called *beaconing* mechanism. The beaconing mechanism for the network partitioning recovery system not only provides the information about neighbor's existence, but also the location information of the neighbor generating the `BEACON`. Since the location information is also needed for other system components and already carried by most IP packets as described in section 5.5, the same data structure is used to transport this information within the `BEACON` messages.

The information from the `BEACON` messages is stored by the node into a neighbor table. The neighbor table maintains a list of all current neighbors of the node. The actuality of this information depends on two parameters: the beaconing interval and the time to live. The beaconing interval is the time between two `BEACON` messages broadcast by the node, and time to live is the time interval during which an entry in the neighbor table is still valid. If, for instance, it is allowed, that one `BEACON` message can be lost in one time, the time to live has to be equal to the double beaconing interval.

The beaconing interval itself should be selected as a trade off of several parameters of the network. On one side, the higher the node speed is, the shorter the beaconing interval should be. On the other side, too short beaconing intervals can produce too much overhead in the network. It is especially critical for networks with a high load and a high node density. The collisions caused by a simultaneous broadcasting of the `BEACON` messages by multiple nodes can lead to a dramatical deterioration of the network performance.

For this work, the beaconing interval is selected with respect to the following circumstances: the position of a node should not change more than on 10 % (expressed in meters) of the communication radius of the node during the beaconing interval. So, if a reference communication radius R_{ref} is 300 m, and the maximal speed V_{max} of the nodes is about 20 m/s, then the beaconing interval T_b can be calculated as $T_b = \frac{R_{ref} \times 10\%}{V_{max}} = 1.5$ s.

In spite of the fact that a routing approach presented in the network can utilize some beaconing mechanism already, the system includes its own beaconing mechanism to stay independent of the used routing scheme.

Of course, there are many different ways to optimize the beaconing mechanism, and the first optimization problem is the selection of the proper beaconing interval by which the network is not overloaded but the neighbor information is still actual enough. The second way is to piggyback the information from the `BEACON` messages to short IP packets, that can be efficiently sent in highly loaded networks. There are more sophisticated methods as, for example, presented in [IPD11] which try to utilize the stochastic parameters of the network, like the nodes density in each point of time with aim to adopt the beaconing intervals to the current network situation. Regarding the proposed system, the beaconing mechanism can be integrated with `HELLO` messages of the routing mechanism. However, it makes the system tightly bound to this routing mechanism.

5.4 Location System

Another vital part of the network partitioning recovery system is the location system available in the network. The location system is necessary for the placement position estimation for additional nodes used for the network recovery and for the navigation of the additional nodes. In this section, a short overview over the most commonly used localization approaches and techniques is given together with the description of used coordinate systems and transformations and with some optimization of distance calculation implemented for the system.

5.4.1 Common Approaches for Node Location

Since the location systems are extremely important in a wide range of human activities, this subject is being actively investigated by scientists and researchers. In one or the other form, location systems find an application in the industry, the military, the emergency and the rescue services, on the transport, and also for the gaming and the entertainment, as well as in many other areas. Constantly, new methods for localization accuracy improvement are investigated and developed, that leads to emergency of a large number of different location systems present in literature.

The authors of [BZGV08] propose to divide the location systems into two groups according the approach type used by the system: mapping (or fingerprinting) and geometric and statistical. The fingerprinting methods are based on comparison of local measurements of signal parameters with the preserved values from a database. Such a database, for example, can contain the signal strength measurements from a set of

GSM base stations for each point of some area. However, the fingerprint methods have two major disadvantages. Firstly, the higher the spatial resolution of the measurement points and the larger the area is, the more measurement data should be collected and maintained in the database. This costs much effort by measurements and by the database lookup during the localization. Secondly, this method is only applicable for a more or less static scenario since a minor change in the environment can make all collected date useless for the localization, and the lookup data should be measured again at least for a part of the area.

A most commonly used parameter for the measurements is the Received Signal Strength (RSS) value, which is measured by the device being localized for some reference transmitter (another network node, or an element of the infrastructure). However, an appropriate radio signal propagation model is needed in order to allow distance estimation based on RSS.

For example, the Environment-Aware RSS-Based Location Estimation (EARBALE) scheme proposed in [TAGT08] uses the premeasured data to adopt the radio signal propagation model, which is used for RSS-based distance estimation from multiple GSM base stations. The collected data is used during the learning process for an artificial neural network which provides the parameters for the propagation model. After that, triangulation is used for the position estimation of the mobile terminal.

The group of geometric and statistical methods utilizes different parameters of the physics and geometry of radio signal propagation combined with different statistical signal propagation models. The most common signal properties used for the localization are based on the features from the following thee domains: signal strength, direction and time. Some of these features are listed below:

- **Received Signal Strength (RSS)** – one of the most popular parameters, used for localization. RSS makes possible to estimate the distance between the transceiver and the receiver (e.g. [TAGT08]), also in fingerprinting approaches as mentioned above.

- **Time-of-Arrival (ToA)** is the time which the radio signal needs for travelling from a remote transmitter to the receiver. Of course, synchronization of the transceiver and the receiver is needed to calculate the propagation time (e.g. [GCWI08, TPQ08]).

- **Time-Difference-of-Arrival (TDoA)** method in contrast to ToA uses the time difference of signals received from multiple receivers or transmitted by multiple transceivers (e.g. [BR08]).

- **Angle-of-Arrival (AoA)** is measured by a receiver equipped with multiple antennas: antenna array or multisector antenna. Measuring either time or signal strength parameters among different elements of a compound antenna makes it possible to estimate the angle the signal was received from (e.g. [TPQ08]).

From the measurements listed above, either the distance between a transceiver and a receiver or the direction from the transceiver to the receiver can be estimated. Combining the estimations it is possible to calculate a relative or an absolute position of the nodes using the following techniques:

- **Triangulation**. For the localization, the measurement data of AoA collected from at least two reference points is required. In order to localize the node, the distance between the measurement points should be known (e.g. [ECC03]).

- **Tri- or Multilateration**. For the localization, the difference of distance measurements from three or more known locations for the mobile node to be localized is used (e.g. [HSZ+12]).

- **Hop count**. While calculating the number of hops between nodes, it is possible to make a rough estimate of the relative position of the node. However, this method is not applicable for networks with dynamically changing topology (e.g. [WLRS05]).

There also are researches aimed to improve the localization accuracy using some additional a priory known information, like known distances between some nodes of a wireless sensor network. Examples of localization improvement schemes can be found in [AST10] and [ASM+11]. The results presented in these papers demonstrate a significant accuracy improvement for the nodes localization.

Once a relative position of the node is estimated, the node coordinates should be calculated within a common coordinate system. The process in which the nodes of a network negotiate the coordinate system is called coordinate system registration [BT05]. Typically, a coordinate transformation may include shift, scale and rotation. The selection of the coordinate system depends on the used scenario. For an indoor localization, it is reasonable to define a coordinate system bound to certain points in a building or in a room. For the outdoor scenario, it is typical to use the global coordinate system, where the coordinates are expressed in degrees of latitude and longitude with possible specification of altitude for the localization in 3D space.

For the placement of additional nodes, the network partitioning recovery system presented in this work uses global coordinates. However, it is inconvenient to calculate the distances in terms of latitude, longitude and altitude, therefore all the distances calculated in the system are measured in meters. The distance calculation using geographical coordinates is explained in sections 5.4.2.1 and 5.4.2.2.

5.4.2 Outdoor Location Systems

For the location of objects on the Earth surface, satellite-based location systems are used. Two satellite systems are available for the global navigation now: the Global

Positioning System (GPS) [GPS12] from USA is de facto standard for the navigation, and the Global Navigation Satellite System (GLONASS) [GLO12] from Russia is established last years. The navigation microchips, which are able to work with both systems, are already available and deployed in some models of smartphones. Since the location quality is directly depending of the number of visible satellites, using both systems at the same time improves the availability and the accuracy of navigation.

Other navigation satellite systems are being under active development: Galileo [Gal12] in European Union, and BeiDou (Compass) [Chi11] in China.

5.4.2.1 Coordinate system

Another important question of the global navigation is the used coordinate system. Since the Earth surface is not an ideal sphere, there are several elliptic models which define a set of parameters, describing the geometry of the Earth surface. This work relies on the World Geodetic System 1984 (WGS 84) ellipsoid model [EUR98] which is used as the reference coordinate system by GPS. The WGS 84 standard was initially published in 1984 and last revised in 2004. The standard defines an accuracy of ±1 m horizontal and ±2 m vertical.

An important task which should be solved is the distance calculation between two points given by their latitude and longitude. For this purpose, the WGS 84 standard provides an algorithm for the calculation of the length of a curve connecting these two points on the surface of the WGS 84 ellipsoid. However, the algorithm needs much computational effort on a mobile node, and it is not efficient to use it in approaches which rely on multiple distance calculations.

5.4.2.2 Distances Calculation Optimization

To reduce the computational effort needed for distance calculation used by the algorithms of the network partitioning system, the distance calculation can be linearized within the area covered by the network. Since the network coverage area of a MANET is small in comparison to the Earth surface, it can be approximated by a plane.

For this work, the following linearization algorithm is defined. For a set of points participating in multiple distance calculation, the algorithm takes minimal and maximal values for longitude and latitude: lat_{min}, lat_{max}, lon_{min} lon_{max} as shown in figure 5.5 and calculate two distances using the WGS 84 distance calculation algorithm:

$$\begin{aligned} d_{lat} &= distance(A, B); \\ d_{lon} &= distance(C, D) \end{aligned} \tag{5.1}$$

where the point A, B, C and D have the following coordinates:

$$A = (\frac{lon_{min} + lon_{max}}{2}; lat_{max}); B = (\frac{lon_{min} + lon_{max}}{2}; lat_{min});$$
$$C = (lon_{min}; \frac{lat_{min} + lat_{max}}{2}); D = (lon_{max}; \frac{lat_{min} + lat_{max}}{2}). \quad (5.2)$$

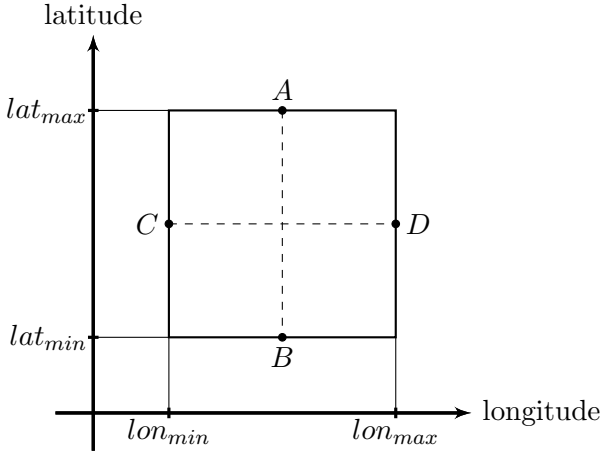


Figure 5.5: Reference points for the linearization

Then, the linearization coefficients are

$$k_{lat} = \frac{d_{lat}}{lat_{max} - lat_{min}};$$
$$k_{lon} = \frac{d_{lon}}{lon_{max} - lon_{min}} \quad (5.3)$$

With these coefficients, the distance between two arbitrary points X and Y from the area can be easily calculated using Euclidean distance calculation:

$$d_{XY} = \sqrt{[(lat_X - lat_Y)k_{lat}]^2 + [(lon_X - lon_Y)k_{lon}]^2} \quad (5.4)$$

Since the area specified by pairs of minimal and maximal longitudes and minimal and maximal latitudes is not a rectangle but a trapeze on an elliptic surface, the linearization produces some inaccuracy which grows near the poles. The diagram on figure 5.6 shows the linearization error during measurement of the length of the diagonal for an area of one degree of latitude by one degree of longitude in size for different latitude values. The diagram indicates that the error does not exceed 5 meter in range from 0 to 85 latitude degrees. For comparison, the accuracy of a modern GPS device is about 4 to 6 meters (7.8 m 95 %) [Gri08]. Moreover, the linearization error value corresponds to

a distance more than 100 000 m (157 000 m at the equator and 112 000 m near the poles), also does not exceed 0.004 %.

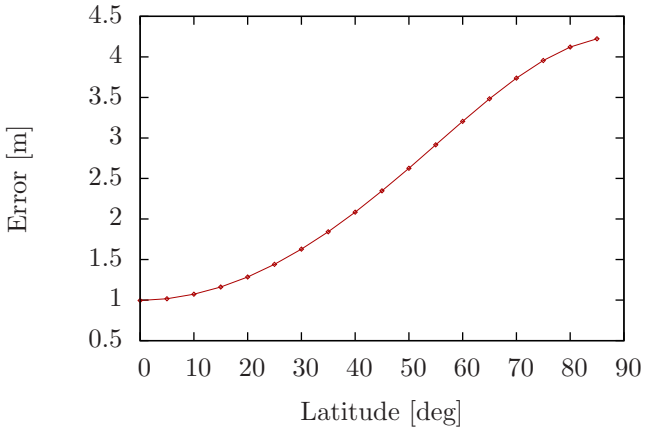


Figure 5.6: Linearization error by the distance calculation for different latitudes

Additionally to the distance measurements, the linearization simplifies the calculation of the coordinates for a point having a specified distance from the given point. Such a calculation is used for position estimation for the placement of additional nodes.

5.4.3 GPS in Click

In Linux, the GPS Daemon (GPSD) [R+12] is responsible for the interaction of applications with GPS devices. The daemon GPSD is running in background as a system service and is available for the applications via a socket-based interface. GPSD also provides an API to simplify the interaction of the application with the service.

Since the GPS data should be available for Click elements used by the network partitioning recovery system, the *GPSReciever* is implemented and provides this functionality. The element starts a background thread communicating directly with the GSPD service which polls the GPS device continuously. The data of each GPS fix is stored directly within the class of the *GPSReciever* element and is accessible via method interface for other Click elements requiring this. For this data, a simple structure consisting of three floating point number fields is defined: one field for each of *latitude*, *longitude* and *altitude* values. If a new GPS fix is not available during a long time, the *GPSReciever* element keeps the last known one.

For test and simulation purposes, the *FakeGPSDevice* is also implemented and can be used instead of the *GPSReciever*. Via handle interface of the *FakeGPSDevice* element it is possible to assign to the node some fake coordinates, so that any needed network topology can be simulated. In section 6.2.2 the usage of the *FakeGPSDevice* will be discussed in more detail.

5.5 Extended IP Headers

As mentioned above, it is necessary for some of the system components to be able to transport the position information. First of all, it is important for the communication partner tracking mechanism. To avoid the transmission of additional messages, the position information can be integrated into other messages being processed by the network.

The position information consists of the following three components: latitude, longitude and altitude. To reduce the data size to be transferred, the latitude and the longitude values are presented in 4 Bytes floating point number format as in the same time an 8 Bytes format is used for calculations. The inaccuracy of the storing format does not exceed $1 \cdot 10^{-6}$ degrees for the whole range from -180 to 180 degrees. This value corresponds to a 0.112 m distance near the poles where the inaccuracy is maximal. However, the 4 Bytes floating point format is not enough for the calculation due to the error accumulation. The altitude value does not have significant impact on the calculations and therefore it is stored as a 2 Bytes signed integer value.

To avoid modification of the existing protocol implementations, the position information is stored as an option within the IP headers. In the following sections, the option structure for both versions 4 and 6 of IP protocol is discussed. Additionally, the option is added only to short IP packets, which should not be fragmented after the extension.

5.5.1 Integration into IPv4

The option structure used with IPv4 is defined in RFC 791 [RFC81]. The header of an IPv4 option contains the following fields:

- `C` – 1 bit – Copy flag indicates whether the options will be copied into each fragment on packet fragmentation or not.
- `CL` – 2 bits – Option class specifies the class of the option (0 – control, 2 – debugging and measurements, 1 and 3 are reserved for future use).
- `Number` – 5 bits – A number specifying the option's type. Some numbers are already allocated in [C+11], other numbers are free.
- `Length` – 8 bits – The total length of the option including the header.
- `Data` – If needed, the option can include some additional data of variable length.

The option containing the position information is shown in figure 5.7 together with the corresponding IPv4 header. The fields of the IPv4 header affected by the extension are:

- `IP Header Length (IHL)` – the length of the IP header counted in fours of bytes, will be increased by 3 (for 12 Bytes of the option).
- `Total Length` – the length of the whole IP packet should be increased by 12 Bytes.
- `Header Checksum` should be recalculated with respect to the changed fields and to the added option.

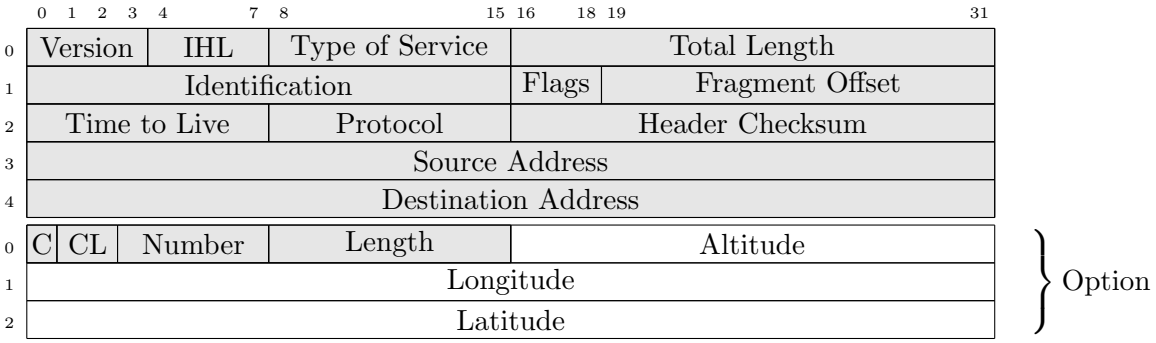


Figure 5.7: Coordinate option within an IPv4 header

The option header is built as follows: the copy flag `C` is set to 0 so that the option will not be copied on fragmentation. For the option class `CL` field the following classes are defined: 0 – control, 1 and 3 – reserved, 2 – debugging and measurements. For the geographical option, the debugging and measurements option class is most suitable (value $10_2 = 2$). The option `Number` is selected as 27. So the three fields build together one byte with value of $5B_{16}$ in hexadecimal format. The `Length` byte is set to $0C_{16} = 12$.

5.5.2 Integration into IPv6

The IPv6 packet structure differs significantly from the IPv4 packet structure. The most important difference is that the options within an IPv6 header should be included into an extension header. At the moment, two extension headers are defined to carry the options: the Hop-by-Hop Options header and the Destination Options header [DH98]. The options placed into the Hop-by-Hop Options header are examined by each node along the packet's path, and the options contained in the Destination Options header are ignored by all intermediate nodes and evaluated only by the destination node. For this work, the Destination Options header is more suitable, since the position information should be accessible only at the end points of a communication. However, if one of both extension headers is already present in the packet, it is rational to add the option containing the position information to the existing extension header.

Within the IPv6 header itself, only two fields are affected by the extension: the `Payload Length` field and the `Next Header` field. Since all the extension headers are included into `Payload Length`, the length value should be updated with respect to

the added/changed extension header. The `Next Header` field should contain the type number of the following extension header. So if the Hop-by-Hop Options header is used, this value is set to 0, and for the Destination Options header to 60 (decimal).

For an IPv6 packet, the recommended extensions header order defined in section 4.1 of RFC 2460 [DH98] should be taken into account. The recommended header order is: the IPv6 header, the Hop-By-Hop Options header, the Destination Options header, and other extension headers.

An example of the extension header containing the geographical option is shown in figure 5.8. Please note that the presented extension header structure is the same for both the Hop-by-Hop Options header and the Destination Options header. The only difference is the value of the `Next Header` field of the directly preceding header. The `Next Header` field of the extension header itself contains the number of the next extension header or of the upper layer protocol. The `Header Length` field contains the length of the extension header in 8-byte units, not including the first 8 Bytes. In the example, presented in figure 5.8 this value is one, so that the total extension header length is 16 Bytes. For the padding, the special PadN option (option type 1) with 0 Bytes of padding data is used.

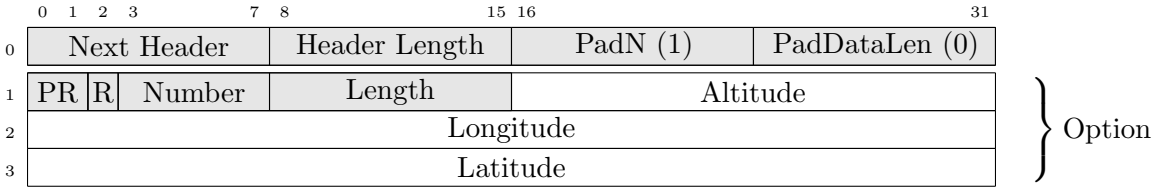


Figure 5.8: Coordinate option within an IPv6 hop-by-hop extension header

The geographical option structure used with IPv6 is also slightly different in comparison to the corresponding structure for IPv4. The `PR` field defines how the IP packet containing this option will be processed by the node which does not recognize this option type. For the geographical option this value is set to 0 to indicate, that the option should be ignored if it is unknown for the host. The next bit flag `R` indicates whether the option may affect routing or not and is set to 0 for the geographical option. The option `Number` was selected to be 27, the same as for IPv4. The option type built by three fields described above has value of $1B_{16}$ for the geographical option.

Obviously, the transportation of the geographical information produces four bytes more overhead (16 Bytes extension header with geographical option in comparison to 12 Bytes in IPv4) for IPv6 than for IPv4. This value stays constant regardless of whether the packet already has an options extension header or not, since it should be aligned to the border given by an integer multiple of 8 bytes anyway.

Another way to transport the position information within the header of an IPv6 packet is to define a separate extension header carrying this information. In spite of the fact that the total data size in this case is 12 Bytes (1 Byte per the `Next Header` and the `Header Length` fields and 10 Bytes of location information), the extension header needs 4 Bytes of padding to be an integer multiple of 8 bytes anyway.

5.5.3 Alternative Implementations

Two alternative ways to implement this functionality for a Linux platform were examined and compared for this work. The first way is the implementation in a separate Linux Kernel Module (LKM) using NetFilter API [MR+12] provided by the Linux kernel. In this case, each packet traversing the protocol stack of the node can be intercepted by a hook function. The second way is to implement the extension functionality in Click elements and integrate it into the Click router script.

The performance evaluation of both implementations is presented in table 5.2. For the evaluation, the following three scenarios were selected:

- transfer of full size packets over one second (peak traffic) using `iperf` utility;
- transfer of full size packets over ten seconds (normal broadband traffic) using `iperf` utility;
- transfer of maximal possible number of short packets (110 Bytes) over ten seconds using flooding ping.

Table 5.2: Performance comparison of LKM and Click implementations for IPv4 header extension

Test	**LKM, MBit/s**	**Click (w/o ext.), MBit/s**	**Difference**
Full size packets, 1 s (`iperf -t 1`)	17.35	17.28 (17.36)	-0.40 %
Full size packets, 10 s (`iperf -t 10`)	16.42	16.02 (16.33)	-2.44 %
Small packets, flood ping, 10 s (`ping -f -w 10`)	12.51	9.17 (9.55)	-26.70 %

The experiment was repeated ten times for each scenario and for each extension method. In the table, the average values of ten experiments are shown. The Difference column shows the performance degradation for the Click implementation in comparison to the LKM implementation in percent.

As follows from the table, the Click implementation shows a significant performance reduction only during the transmission of a large number of short packets while the performance reduction in first two tests is insignificant. Moreover, the measurements indicate that most of the time is spent not on the extension of the packets itself but on the processing of the packets in Click (compare the values from the Click column inside and outside the brackets; the value in brackets corresponds to the packet processing in

Click without any extension). However, since all other components are implemented in Click, the Click implementation was also selected for the packet extension.

5.5.4 Click Implementation

Since the realization of the network partitioning recovery system presented in this work is based on version 4 of IP protocol, only the IPv4 implementation of the IP header extension is considered here. The part of the Click router graph responsible for the IP packet extension is shown in figure 5.9.

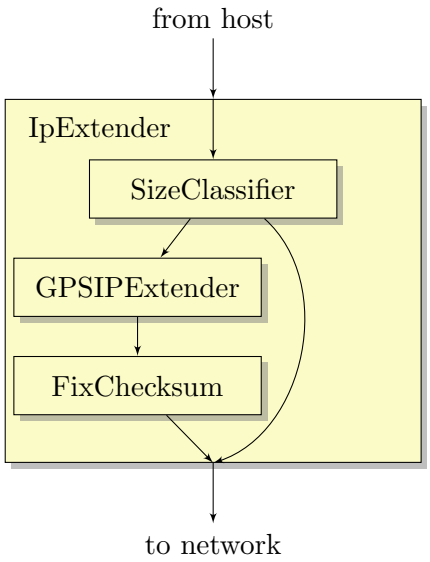


Figure 5.9: Extension of IPv4 headers in Click

The packet extension is used in several places of the node configuration script. Firstly, the packets from the host applications (usage plane traffic) and secondly the packets generated by the Click script itself (repair plane traffic). From this reason, Click elements responsible for the packet extension are incapsulated into the *IpExtender* compound element. The packets produced locally are classified firstly by the *SizeClassifier* so that the big packets are passed through the *IpExtender* element unchanged. Then, the other packets are extended with geographical information directly by the *GPSIPExtender*. After the extension, the checksum of the packet should be also recalculated within the *FixChecksum* element. Finally, all the packets are forwarded to the next element in the network direction.

5.6 Nearest Nodes Search

According to the system architecture described in the previous chapter, the network partitioning recovery system is capable of estimating the placement position for addi-

tional nodes in both situations with and without the availability of the global knowledge about the network topology. In this section, the solution methodology based on the Nearest Nodes Search (NNS) approach, which is used if there is no global knowledge available, is presented.

The solution idea behind the NNS approach is based on the following presuppositions. Let's assume that the network partition B becomes isolated from the partition A (s. fig. 5.10). Obviously, to reconnect the partitions additional nodes should be placed between them, and the exact placement positions are determined by the nodes belonging to the perimeters of A and B. Theoretically, the optimal placement positions are arranged on the line connecting the two closest perimeter nodes from both partitions A and B. However, it is not possible for the partition A to determine which nodes belong to the perimeter of the partition B after the network was split and vice versa. Remember, the system shows reactive behavior and does not attempt to predict where the partitioning can take place.

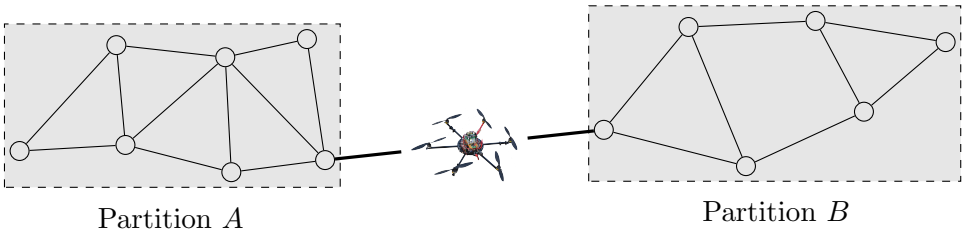


Figure 5.10: Two network partitions reconnected by an additional node

Thereby, the following information is available for or can be collected by the partition A. Firstly, the last known position of a communication partner node before the partitioning is available due to the communication partner tracking mechanism. So, this information can be used for a rough estimation of the direction from A to B. Secondly, it is possible within the partition A to find its own perimeter nodes which are closest to the last known position of the communication partner node from the partition B and eventually also closest to the whole partition B. This search is the task of the NNS procedure.

The NNS procedure can be implemented in several ways. The most promising approach for searching the nodes closest to a given point is based on geographical routing or geocasting. Of course, such nodes can be also found by traversing the whole network using one of the graph traversing algorithms. In addition, the search can utilize some heuristic methods like random search at which the priority is given to the nodes which are closer to the destination point given by the last known position of the communication partner. All three mechanisms are presented in the following sections.

5.6.1 Geographic-Aware Routing and Geocasting

There are many projects on location-aware routing. A classification of MANET routing algorithms is proposed in [BNSM09]. In figure 5.11, the location-aware part of the

classification is shown.

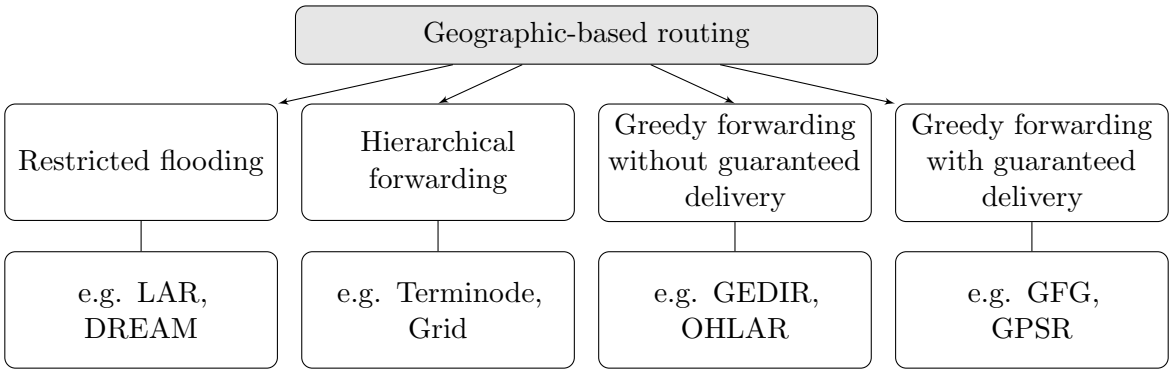


Figure 5.11: Geography-aware routing protocols, based on [BNSM09]

The most simple routing approaches are based on restricted flooding. In this case, the flooding area is limited with respect to the nodes' geographical positions. One of the first approaches is presented in "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks" [KV98]. The main goal of this approach is to reduce the route discovery overhead using geographical information delivered by the Global Positioning System (GPS). It is proposed to define two areas in the network where flooding must be done. These areas are named expected and request zones. The expected zone is defined based on a motion pattern of the destination node, and the request zone includes the expected zone, the source node and all intermediate nodes. Another restricted flooding routing protocol is the Distance Routing Effect Algorithm for Mobility (DREAM) proposed by Basagni et al. in [BCSW98] and evaluated for large MANETs in [BC08]. In this approach, each node proactively maintains a location table about the position of all nodes in the network. The updates periods for the table entries are proportional to the distance between the nodes. Here, a forwarding zone is determined by the direction to the destination node and its expected zone defined similar to the expected zone in LAR. The forwarding zone represents an angle whose vertex is at the source node and whose sides are tangent to the expected zone estimated for the destination. In DREAM, only nodes' coordinates are being exchanged instead of complete link state or distance vector information.

The next group of the location-aided routing protocols can be classified as hierarchically forwarding protocols. One of them is the Terminode routing algorithm which is proposed in [BLG05]. This approach combines a location-based routing method with a link state-based mechanism. The location-based routing method is used when the distance to the destination is long and the link state routing is used on a short distance in a local environment. The Terminode routing uses a special form of restricted local flooding. For message forwarding, this approach utilizes a concept of geographical anchor points imagined by the source in the direction of the destination.

The other example of a location-based protocol with hierarchical data forwarding is Grid [LJD+00]. The main assumption is that the source node knows the geographical

position of the destination node. The packets are being forwarded through the network based only on the geographical information of the destination. Each node in the network maintains a table with identities and geographical position of its neighbors. The whole network is split into grids which include a "location server". This "location server" is elected for each grid and stores information about current coordinates of the nodes within its grid. Thereby, the message to a node far away is being forwarded by the "location servers" and is delivered to the destination at the last grid.

The next group of protocols relies on greedy forwarding. One of the protocols in this group is Geographic Distance Routing (GEDIR) [SL99, WH07]. In GEDIR, the next hop is chosen as the nearest neighbor node to the destination, and a Routing Request (`RREQ`) message will be forwarded to this node. Nevertheless, a route discovery is not guaranteed even if the route exists. This can happen because of possible "dead ends" when all neighbors are farther from the destination than the current node. The possibility of "dead ends" grows with decreasing nodes density. In the classic GEDIR approach, the neighbor location information is collected trough broadcasting of beacon-messages. A Non-Beacon GEDIR modification was proposed in [WH07]. In this case, the neighbor location information will be requested only in reaction to the RREQ-message, thus reactively.

Another example of greedy routing without guaranteed delivery is One-Hop LAR (OHLAR) and its extended version – improved OHLAR as proposed in [LL08]. It is a special case of routing for MANETs which implies the existence of a WiMAX infrastructure. In this approach, the routing control information, such as the coordinates of the nearest neighbors and the nodes' position, is being exchanged over a WiMAX infrastructure, and the data are transported as in a MANET via multi-hop communication. In the improved OHLAR, the authors propose a solution to the ping-pong effect caused by "dead ends": If there are no closer neighbors to the destination, the message will be sent back to the previous node and this node tries to send this message to some other closest neighbor to the destination.

There also is a number of greedy protocols with guaranteed delivery. One of them is the Greedy-Face-Greedy (GFG) protocol presented in [BMSU99]. In this approach, the whole network is modeled as a unit graph. The unit graph is a network topology graph in which edges are present only if a bidirectional link between the nodes exists. Delivery can be guaranteed as long as the unit graph is static and connected during all time of message forwarding. This approach does not require additional memory on the nodes and adds only a constant amount of information to the packet. No packets are duplicated in the nodes. The approach finds a connected planar graph based on the unit graph and starts some routing algorithm on it. GFG supports geocasting, a multicast to some geographic area, and broadcasting. The main problem of GFG is that it does not work, by design, with a dynamic topology and must be extended for this purpose.

Another example of greedy protocols with guaranteed delivery is the Greedy Perimeter Stateless Routing (GPSR) proposed in [KK00]. Just like the previous examples, this

kind of routing needs the information about immediate neighbors only. If the routing reaches the "dead end" region, the algorithm recovers by routing around the perimeter of this region based on a planarized network graph. To map the nodes' addresses to the location, each node maintains the neighbors table. This table is being filled in two ways: first, the nodes periodically broadcast beacon-messages with their address and coordinates. To minimize collision probability, the beacons are sent with a jitter of about 50 % of the inter-beacon interval. Secondly, to minimize the control overhead, the coordinates information of the currently forwarding node is added to each forwarded packet. So, each packet can play the role of the beacon message and due to this the control overhead can be significantly reduced in active parts of the network.

As shown before, there is a large amount of location-aware routing protocols. Some of them assume a hierarchical composition of the network, which leads to extra communication, memory and computation overhead. These approaches also convceived some solutions interesting to this work. For example, greedy algorithms offer a simple forwarding solution based only on the immediate neighbor information and propose some methods to avoid the ping-pong effect in "dead ends" of the network topology. These algorithms do not need to many communication overhead but provide a reliable method for finding the routes to the destination. Furthermore, some of these algorithms (for example, GPSR) can even be optimized and extended to provide a geocasting functionality.

Geocasting A special case of geographical routing is geocasting. The task of geocasting is message delivery to the nodes within a specified geographical region. Typically, it combines a geographic routing protocol to forward the message to the geocast region (GCR), and a flooding protocol to disseminate the message between the nodes within the GCR. There are also some other approaches being discussed later.

In [JC02], the authors divide the geocasting protocols into two classes: data transmission oriented protocols and route-creation oriented protocols. The protocols of the first group flood the GCR with the message, and the protocols of the second group try to find routes to the nodes within the GCR. Examples for the transmission oriented protocols are the Location-Based Multicast algorithm (LBM) [KV99], a geocasting protocol based on the Voronoi diagram [SRL03], and GeoGRID [LTS00], a geocasting-aware extension for the GRID routing protocol [LTS01]. The examples for the route-creation oriented protocols are GeoTORA [KV00], an extension of a Temporally Ordered Routing Algorithm (TORA) [PC01], and an adaptive mesh-based geocast routing algorithm, the so called Geocast Adaptive Mesh Environment for Routing (GAMER) [CL03].

The authors of "Performance Comparison of Geocast Routing Protocols for a MANET" [YKC04] propose another classification for the geocasting routing protocols. They divide the protocols into three groups: flooding-based protocols, routing-based protocols, and cluster-based protocols. Unlike the classification above, here is taken into account how these protocols forward the message from the source node to the GCR. The protocols

of the first group use some kind of flooding to deliver the message to the destination region. Examples for this group are the LBM and the Voronoi diagrams based routing. To the second group belong the protocols which try to find some route or routes to the destination region. Such protocols are, for example, GAMER and GeoTORA. In the third protocol group, a hierarchical logical network structure is considered. In this case, the whole network will be structured into clusters which can be a grid, as in the GeoGRID routing protocol, or a hexagonal cell, as in the obstacle-free single- or multi-destination geocasting protocol [CCT03]. In each class, a cluster head must be elected whose main task is inter-cluster routing.

The authors provide four studies of performance evaluation for each of the three protocol groups. The four studies are: network models with variable nodes density, with presence of additional traffic or congestion, with several mobility levels, and with a combination of these factors. For these studies, one protocol from each group has been selected: LBM as a flooding-based protocol, GAMER as a routing-based protocol, and GeoGRID as a cluster-based protocol. GAMER showed the lowest overhead in most experiments, while GeoGRID has been presented as the most robust protocol, especially in dense networks. LBM, the simplest protocol, demonstrated low performance in general, but it can be equivalent in some special cases. Below, the presented algorithms are described in detail.

One of the first approaches for geocasting is the Location-Based Multicast algorithm (LBM) proposed by Vaidya et al. in [KV99]. The main goal of this approach is the reduction of the forwarding region for multicasting to decrease the number of messages. The goal region is called a multicast region which can be defined as a closed polygon or a circle. The message will be flooded within the multicast region. To restrict the flooding zone outside the multicast region, a forwarding zone will be defined by the source node. By definition, the forwarding zone includes both the source node and the multicast region. There are multiple ways to define the forwarding zone. Thus, a trade-off between the accuracy of multicast delivery and the produced overhead must be found. The authors propose two schemes to define the forwarding region. Firstly, the forwarding zone is defined as a minimal rectangle which includes the source node and the multicast region. To optimize the delivery rate and the generated overhead, the forwarding zone can be extended or reduced with a parameter δ. Secondly, each node calculates the forwarding zone adaptively. For this purpose, the distance between a sender and the center of the multicast region will be calculated. The node transmits the message if its distance to the multicast region is shorter relatively to the distance from a previous hop. Otherwise, the message will be discarded.

Another geocasting protocol is GeoTORA proposed also by Vaidya et al. in [KV00]. This protocol combines the unicast routing approach to the GCR with flooding within the GCR. The unicast routing approach, used by GeoTORA, is the Temporally Ordered Routing Algorithm (TORA) [PC01] which uses a link reversal algorithm and builds a destination-oriented directed acyclic graph (DAG) for each destination. GeoTORA extends TORA with “anycast”, the message delivery to any one node of the group. The

nodes within the GCR build the anycast group, and GeoTORA maintains one DAG to each GCR. Thus, in the first phase, the message will be routed to one node of the geocast group (anycasting), and, in the second phase, will be flooded within the GCR. The main problem by flooding within the GCR is that the forwarding fails if the GCR has isolated partitions which are still connected within the whole network. To avoid this problem, several other protocols are proposed.

One geocasting protocol, which can deal with isolated partitions within the GCR, is the Obstacle-Free Geocasting Protocol (OFGP) proposed in [CCT03]. With OFGP, the whole network will be clustered into hexagonal cells. Within each cell, a cell manager is elected as cluster head. The cell manager is responsible for inter-cell communication and periodically broadcasts an "existence" message. The cells without any "existence" messages within a predefined period will be marked as an "obstacle". Due to the "existence" messages, the network can detect the loss of connectivity within the GCR. Two phases in OFPG are presented: a reaching phase and a broadcasting phase. In the first phase, the network attempts to send the message to any host within the GCR, and, in the second phase, the message will be broadcast to all members of the geocasting group. The main issues of this approach are extra overhead generated by forming the hierarchy and cell manager election, and the necessity to transform the coordinates to provide an addressing within a hexagonal cellular structure.

Another approach, proposed in [SRL03], uses a Voronoi diagram built on the local neighborhood of the node to select a next hop in the destination direction. The Voronoi diagram of n points in a plain splits the plain into Voronoi regions associated with each point. By definition, the Voronoi region of point N consists of all the points in the plain which are closer to N than any other Voronoi points. Each region is a convex polygon (possibly unbounded) determined by bisectors of N and all other Voronoi points. As a next hop, the node will be selected whose Voronoi region includes most part of the destination region. The authors propose this approach as an extension of the GEDIR [SL99, WH07] routing protocol and call it V-GEDIR.

The GAMER [CL03] protocol bases on three concepts: source routing, forwarding zones, and meshes. By source routing, each packet carries the complete route within itself. GAMER defines three types of forwarding zones: the entire network (network-wide flooding), a corridor (sides of the corridor are built by two parallel lines convex to the GCR and the source node between them), and a cone with vertex into the source node and sides convex to the GCD. GAMER selects each type adaptively according to the current network topology and the packet drop rate. In GAMER, a mesh is defined as a set of multiple paths within the forwarding zone. As soon as the first path is found, GAMER tries to find a next path within a smaller forwarding zone.

However, LBM, Voronoi diagram or Mesh-based approaches, and some of the geographic routing protocols have a common problem: the local maxima problem, which is illustrated in figure 5.12. In this scenario, a path from the source node S to the GCR exists in the network, but will not be found by the approaches which forward the messages to the neighbor closest to the destination . From S's point of view, the closest

neighbor to the destination is M', but to be successful, the routing must go into the opposite direction via node M. Geocast routing protocols, which try to avoid the local maxima problem, will be discussed next.

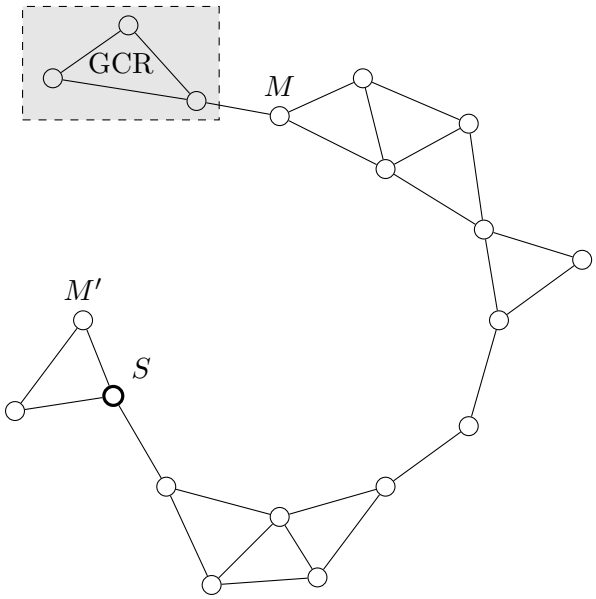


Figure 5.12: Local maxima problem by the geographic routing

Some solution ideas in this direction were proposed by Seada and Helmy in [SH04] and [SH06]. They present two protocols for geocast routing: Geographic-Forwarding-Geocast (GFG) and Geographic-Forwarding-Perimeter-Geocast (GFPG). Both algorithms utilize geographic forwarding from the source to the GCR and regional flooding within the GCR. For geographic forwarding, they use greedy routing with perimeter routing to avoid the "dead ends" and the local maxima problem, similar to the GPSR [KK00] routing protocol, which was discussed above. The main goal of this work is the message delivery to all nodes within the GCR. GFG uses a simple flooding within the GCR and produces close-to-minimum overhead in dense networks. GFPG provides delivery guarantee without global flooding in the whole network and any globally available network information if, in the GCR, some obstacles or gaps exist. GFPG uses perimeter routing to transport the messages between groups of isolated nodes within the GCR. Of course, these groups are connected within the whole network. However, GFPG produces more overhead in dense network as GFG. To resolve this problem, one modification of GFPG, called GFPG*, was proposed which utilizes perimeter routing only if the gaps or the obstacles are present within the GCR. To detect the presence of gaps, each node within the GCR divides its radio range region into four sectors. If the node has neighbors within each sector, it is supposed that no gaps are present and no perimeter routing will be used.

An approach, proposed by Jeon and Kessidis in [JK07], utilizes an Ant Colony Optimization algorithm to provide energy-efficient load balancing and multipath routing in the network. They present the Geographic-aware Prioritized Pheromone-aided

Routing Algorithm (GeoPPRA). Forward and backward ants discover multiple node disjoint routes from the source to the nodes of the GCR. These multiple routes are utilized simultaneously per data flow. Two priority levels for the user traffic are defined: latency-critical traffic and latency-non-critical traffic. Two types of the pheromone are involved in the optimization process: energy-pheromone and delay-pheromone. The energy-pheromone corresponds to the energy (battery) levels of intermediate nodes along a path, and the delay-pheromone reflects the latency of the path. For the latency-critical traffic, both pheromones are taken into account, and for the latency-non-critical traffic only the energy-pheromone is calculated. If a link of the path breaks down, the data packet is broadcast within a rectangular forwarding zone and an error message is sent to the source node which will send new ants to discover new paths. Such a behavior increases the delivery rate but produces extra communication overhead.

Song and Lutflyya propose the Zone-based Hierarchical Link State (ZHLS) routing protocol to guarantee the delivery by geocasting in [SL09]. The main goal of this approach is trying to take into account the mobility of the nodes in the MANET. The whole network is split into zones, and the idea is that a zone level topology is relatively stable in comparison to a node level topology. For intra-zone routing, this approach utilizes some shortest path algorithm. However, the zone level topology information must be maintained within the network which causes extra communication overhead in the network. The authors showed that the messages will be successfully delivered in 93 % of all cases.

Some other approach is proposed by Maymi and Rodriguez-Martinez in [MRM09]. The authors define some utility-functions, values of which define the message relaying probability. A relaying utility function is calculated with the following formula:

$$u_{i,t}(m) = e^{-\frac{\alpha p_{i,m} d_i}{b_{i,t}}} \tag{5.5}$$

The relaying utility function value $u_{i,t}$ for the node i at time t is calculated for the message m. Following values are taken into account: a measure for the distance $p_{i,m}$ which is calculated as a distance between the current node i and an imaginary line connecting the source and the destination, a number of neighbors d_i, a battery status value $b_{i,t}$, and some weighting factor α with value of 50. Then, the relaying probability for the message is calculated as a ratio from the relaying utility function value to a sum of the values of the relaying utility function and an ignoring utility function.

To provide obstacle avoidance, a HELP flag is included in each packet. The HELP flag has a value of 0 by default. During routing, if at least one neighbor exists within the forwarding zone, which is built as a cone with sides at $\pm 45°$ to the direction from the current node to the destination, the message with a corresponding probability will be transmitted to those neighbors. Otherwise, the HELP flag will be set, and the packet will be transmitted to another neighbor. If the node receives the packet with the HELP flag set to 1, the value of the relaying utility function will be increased in two steps, and the utility function of ignoring will be not affected. Using of the HELP flag can

provide some interesting results also for other geocast routing protocols.

Some authors only deal with messages delivery optimization within the GCR. For example, Hughes and Maghsoudlou propose an efficient coverage-based flooding scheme within the GCR in [HM06]. The authors assume the nodes have different transmission radii, and the node will retransmit the message only if the following two conditions are fulfilled: the transmission radius of the current node is greater than an average transmission radius of the nodes within the geocasting group, and the retransmission can cover some yet uncovered area. The covered area is defined by a coverage matrix with elements corresponding to square areas defined within the GCR. The authors showed that the communication overhead is significantly reduced in comparison to simple flooding.

Several methods for GPS-free geocasting are presented in the literature. The methods utilize, for example, the radio signal strength and the angle of arrival information together with the topology information to derive some common coordinates system for the network. In [LAOF05], the authors use an indoor location tracking system with the AODV-bis routing – the geocast-enhanced version of AODV [PRD03, PRC03, OF04]. The authors in [MMS06] combine a local positioning system based on one of the variants of the self-positioning algorithms with the LBM geocasting algorithm. This approach calculates the distance between nodes from the time of arrival. Using this information, each node builds its local coordinates system with itself as a center. Then, the local coordinate systems of the nodes will be transformed and combined to the global coordinates system.

There is a number of other geocasting methods which are specific for a concrete environment or some optimization parameters. The authors in [LK06] propose the Geometry-driven Geocasting Protocol (GGP) which deals with multiple GCRs. It builds a tree-like structure with shared paths based on the "Fermat Point" concept. The Fermat Point is a point within a triangle for which the sum of its distances from the vertices of the triangle becomes a minimum. The protocol includes two phases: the greedy routing along the tree and the regional flooding within each of the GCRs. However, no local maxima problem is considered here. Another approach, proposed in [HLP07], deals with routing optimization by mobility pattern based geocast. This approach utilizes a resource reservation algorithm and an improved version of the LBM. The approach seems to be relatively complicated but, however, the problem of local optima is not considered.

All of the location-aware and the geocast routing algorithms presuppose the existence of some mechanism which can map the nodes identities or addresses to its position. One integrated variant of such a mechanism is proposed in [MK04]. The authors define a location service which allows each node to know the positions of other nodes in the network. The nodes hold the location information about all nodes within a location update zone. The location update zone is defined through a radius in a hop count. In other words, each node maintains the information about all its k-hop neighbors proactively. It can be done with a simple beaconing-mechanism when each node

periodically broadcasts a message with its own coordinates and identity and possibly with a list of identities and coordinates of all its neighbors.

Available Implementations Despite of a number of publications about location-aware and geocast routing protocols, only few implementation results are presented in the literature. Two of them deal with geocast enhancements to the variants of the AODV routing protocol AODV-bis [OF04] and AODV-UU [MKHS09]. In both cases, the route request packets (`RREQ`) are extended with location information for the destination region and a distance field which contains the distance from the source to the destination. Instead of network-wide flooding of the `RREQ`, only nodes, whose distance to the destination is shorter than the distance from the source, will forward the `RREQ` packets. In both cases, the location information is used only in the route discovery process. Just one significant difference between both implementations is that, in extension of AODV-bis, a GPS-free location system based on the received signal strength is used, while the nodes in the AODV-UU version are equipped with GPS. However, only simple network topologies are considered in these implementations, and the routing fails if "dead ends" are present in the network.

One interesting report about geographic routing implementations is presented in [KGKS05]. The first goal of the authors was to implement the GPSR [KK00] protocol discussed above. First results showed permanent delivery failures even with a static network topology. The authors supposed that the main problem is the unit graph assumption made in GPSR and many other geographic routing protocols. In the real world, not all radio links are stable because of long and short fading on a radio channel, multipath propagation of the radio signal and some other radio propagation effects which take place in a real environment. So, the assumption about omnidirectional radio range of the nodes and about always bidirectional links between the nodes is no more suitable.

In case of the local maxima problem, GPSR applies perimeter routing. For perimeter routing, a planarized local subgraph is calculated based on the unit graph. The main goal of the planarization algorithm is to remove crossed links without partitioning the graph. The authors showed that, with a local planarization algorithm, it is not always possible to remove all crossed links without subgraph partitioning in the real environment. So, the authors presented three following problems of the unit graph assumption:

1. A link can be removed in the planar subgraph even if it should exist. As a consequence, the planar subgraph can be partitioned.

2. Unidirectional links can be present in a real radio network.

3. Pairs of crossed links can remain in the supposedly planar subgraph.

To resolve these problems, the authors proposed the distributed Cross-Link Detection Protocol (CLDP). In this protocol, before the planarization algorithm starts, the links are probed in a parallel and distributed manner to detect possible link pathologies. Presented results show a significant improvement of message delivery in the network if CLDP is used. The authors supposed that CLDP is also efficient if location errors take place.

The proposed CLDP algorithm seems to be relatively complicated, and one alternative solution should be using a heuristic method as, for example, any kind of the ant colony optimization algorithm.

5.6.2 Greedy Perimeter Stateless Routing

From the geographical routing mechanisms presented above, the GPSR [KK00] routing was selected for one of the NNS implementations with respect to the following criteria. According to the classification shown in figure 5.11, the algorithms of the first two groups (restricted flooding and hierarchical forwarding) produce high overhead due to message flooding in the first case, and due to building and maintaining a hierarchical network structure in the second case. The algorithms of the greedy forwarding group without guaranteed delivery do not solve the problem of "dead ends" and in consequence they cannot be used for the search among all the perimeter nodes.

For the remaining group, an implementation is available only for the GPSR. GPSR is capable of overcoming the "dead ends" due to the mechanism of the perimeter forwarding based on the right-hand rule. Moreover, the proposed implementation was also tested with a real network, and the drawbacks of the mechanism were exposed (s. [KGKS05] mentioned above). However, the problem indicated there can be solved by integration of the neighbor information for the node into its `BEACON` messages as described in section 4.5.

Since the goal of the GPSR within NNS is not the routing itself or message forwarding to the node given by the coordinates, but only the collection of information about the nodes which are closest to this coordinate point, the GPSR-based algorithm finishes in the originating node and collects a list of a given number of the closest nodes identified by the algorithm itself.

GPSR is a distributed algorithm based on local calculations which are made by each node forwarding the GPSR messages. For the calculations only the neighbor information is used. This information is collected by a simple beaconing mechanism in which the nodes provide the neighbors not only with the information about their own existence and their identity (IP address) but also with their own geographical coordinates. So each node is aware of its neighbor nodes and their positions.

The GPSR algorithm uses two forwarding methods: greedy forwarding and perimeter forwarding. The greedy forwarding is used wherever it is possible, while the perimeter

forwarding solves the problem of "dead-ends" when the greedy forwarding does not work. During greedy forwarding, the node directs the message to the neighbor which has the shortest distance to the destination point. If this forwarding method is possible, it gives the best performance during the routing and provides the fastest transmission towards the destination. However, the method fails when the forwarding reaches a "dead-ends".

The perimeter forwarding makes it possible to recover from the "dead-ends" but needs much more computational effort than the greedy forwarding. Firstly, the perimeter forwarding needs to planarize the network topology graph using one of the distributed graph planarization algorithms. Then it forwards the messages along the faces of the planarized graph according to the right-hand rule. The method saves also the points where the packet has entered the current face in order to be able to switch the face to the next one. Also, the method needs to store the point where the algorithm has switched to the perimeter forwarding mode to guarantee that the greedy forwarding does not come to the same "dead-ends" any more.

The authors of [KK00] propose two different graph planarization algorithms: the Relative Neighborhood Graph (RNG) [Tou80] and the Gabriel Graph (GG) [GS69]. For both algorithms the node needs only the information about its one-hop neighbors. Also the graph derived by the RNG algorithm is a subset of the graph generated by the GG algorithm [GS69].

5.6.3 Heuristic Search

The second alternative of the base algorithm for the nearest node search uses the simple Heuristic Search (HS), or random search approach. The initiator node generates a given number N of search packets. Each packet contains the information about the coordinates of the destination point (search goal) and the information about the initiator node. From the point of view of one of the forwarding nodes i (see Fig. 5.13), all its neighbors can be divided into three groups:

1. the nodes which are closer to the destination point as the node itself (within the circle of radius $\overline{iD}$ with center equals to the destination point);

2. the nodes which are from the same half-plane as the destination point (all nodes within the half-plane α except the nodes of the first group);

3. all other nodes (within the half-plane β).

For each group $z = \{\alpha, \beta, \overline{iD}\}$ of neighbors, different forwarding probabilities can be defined as shown below:

$$P_z = \frac{k_z}{k_\alpha \cdot n_\alpha + k_\beta \cdot n_\beta + k_{\overline{iD}} \cdot n_{\overline{iD}}}, z = \{\alpha, \beta, \overline{iD}\} \tag{5.6}$$

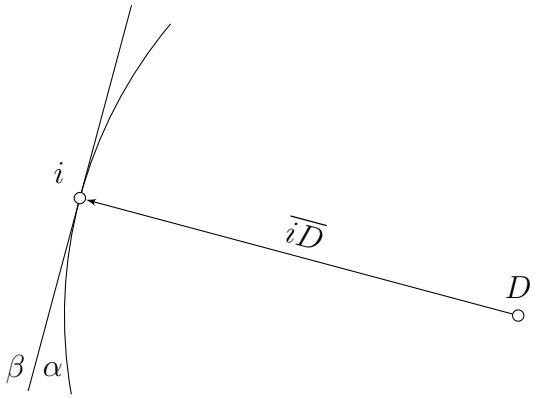


Figure 5.13: Types of the node's neighbors for the Heuristic Search algorithm

Here, P_z is the forwarding probability for each zone z, k are the weighting factors and n is the nodes number for each zone. It is clear from the formula, the values of the weighting factors itself do not matter, but only their ratios are important. Obviously, to provide a better performance, the nodes, which are closer to the destination point D, are more preferable. Therefore, the value of $k_{\overline{iD}}$ should be greater than the values of k_α and k_β, and non-zero values of k_α and k_β provide the opportunity to recover from a "dead-end". Experiments showed that an optimal performance can be achieved if the value of $k_{\overline{iD}}$ it twice greater than the values of k_α and k_β (for example, 10 for zone $\overline{iD}$ and 5 for zones α and β) for a complex topology like the topologies shown in figure 5.14. However, if the network has a regular topology with uniformly distributed nodes, a bigger difference between the coefficients provides a better performance until the difference does not exceed ten times. After that, the performance become worse due to "dead-ends".

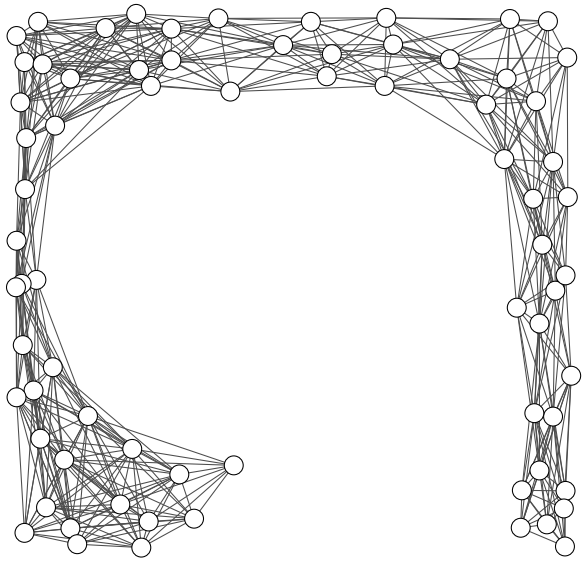


Figure 5.14: Example of a complex network topology with a gap

A response to the search packet will be returned to the originator node if the packet has reached some host within the reference communication radius of the destination or its Time To Live (TTL) has expired. The whole algorithm will be terminated when the originator node collects a given number of the answers. This number is normally lower than the number of initially sent packets, so the result of the algorithm will be constructed from the packets reaching the goal at first.

5.6.4 Distributed Depth-First Search

As mentioned before, the list of nearest nodes can be obtained during the traversing of whole network partition using one of the graph traversal algorithms. Two classical algorithms for graph traversing are known from the graph theory: the depth-first and the breadth-first algorithms. For the network partitioning recovery system a distributed implementation of a traversal algorithm is required.

Since the goal of the application of the algorithms in NNS is to traverse all nodes of the network partitioning, it does not matter which of both algorithms is used. Thereby, the selection criterion is the existence of a simple and efficient implementation. From this reason, an implementation of the Distributed Depth-First Search (DDFS) algorithm, proposed by S. A. M. Makki and G. Havas in "Distributed Algorithms for Depth-First Search" [MH96], was selected. The authors demonstrates also the efficiency of their approach in comparison to others.

In the implementation, two types of messages are used: `forward` and `return` to explore the network graph tree. They optimize classical approaches using the concept of *split points*. A split point is a visited node which has two or more unvisited neighbors. The search message holds the information about its last split point and it, as the return-message, can bypass non split point nodes.

The message, traversing over the network graph, collects information about all nodes within the network partition and additionally holds the IP addresses and positions of the given number of the nodes closest to the destination point.

5.6.5 Problem Specific Extensions

The main goal of NNS is to provide the originating node with information about the positions of a given number of the nodes which are closest to the last known position of one of the node's communication partners. In opposite to classical routing, there is no task to establish a route and to forward the messages. The information collected for the nodes has a specific structure: node IP addresses connected with their geographical coordinates. This information should be delivered to the search originator, and as

follows, the corresponding messages for the different approaches were extended in order to be able to carry this information.

In order to minimize the size of messages, the number of closest nodes collected during the NNS procedure is limited to 3. Each item of this list contains three fields: the IP address, the latitude and the longitude of the candidate node. So in the current implementation using IPv4, each list item takes 12 Bytes (4 Bytes for each field) and the list has a total length of $3 \cdot 12 = 36$ Bytes. Depending on the used algorithm, the size of the complete message is varying. In the following sections, the comparison of the three algorithms presented above and the implementation details will be discussed.

5.6.6 Comparison and Selection of the Different Algorithm

The three algorithms described above: GPSR-based search, DDFS and HS was implemented for the network simulator OMNeT++ 4.1 [VH08] using the additional library INETMANET Framework [Hor08]. The implementation includes the IP header extension and a simple beaconing mechanism. The beaconing interval was implemented using the broadcast `BEACON` messages sent out every three seconds. However, the beaconing interval has no influence on the simulation results with a static scenario, if the NNS procedure is started not in the very beginning. The rare `BEACON` message broadcasting saves also the computational effort needed for the simulation.

The GPSR implementation used for simulation is based on the implementation proposed by B. Karp in [Kar00] with two different planarization algorithms RNG and GG from [KK00]. The HS implementation uses a uniform random number generator for the selection of the next hop for message forwarding. The forwarding probabilities were chosen so that the probability to forward the message to the neighbor which is closer to the destination point as the node itself is two times higher than the forwarding probability for other neighbors. (s. figure 5.13 and formula 5.6 on page 76). Finally, the base for the implementation of the DDFS algorithm is taken from the original paper [MH96].

For comparing the algorithms, the following scenario was defined: the simulated network is located within an area of $1,000 \times 1,000$ m. Two nodes have fixed specified positions: the search originator node and the node within the reference communication radius R_{ref} from the search destination point given by coordinates $(1000; 1000)$. All other nodes are distributed uniformly over the area with the restriction that the network is still connected. Thereby, at least one nearest node should be found by the NNS algorithm. For each number of nodes from 10 to 100 with step 10, ten different topologies were generated. The simulation was repeated 12 times for each of the 100 different topologies grouped by the number of nodes.

As a result, the dependency of the average packet count from the overall number of nodes was determined. Only the packets generated by the search algorithm were taken

into account without consideration of the packets generated by routing and beaconing mechanisms. This dependency is shown in figure 5.15 a). Similarly, the dependency of the searching time from the number of nodes is shown in figure 5.15 b). From the figures, it can be seen that the results collected for the HS algorithm contain a significant random component. The HS method produces much more overhead in comparison to the other algorithms and will not be considered anymore. The GPSR algorithm with the GG planarization demonstrates the best result concerning the produced overhead: the number of the transmitted messages does not grow significantly with the number of nodes in the network. All the algorithms (GPSR with RNG, GPSR with GG and DDFS) take almost the same time for the searching.

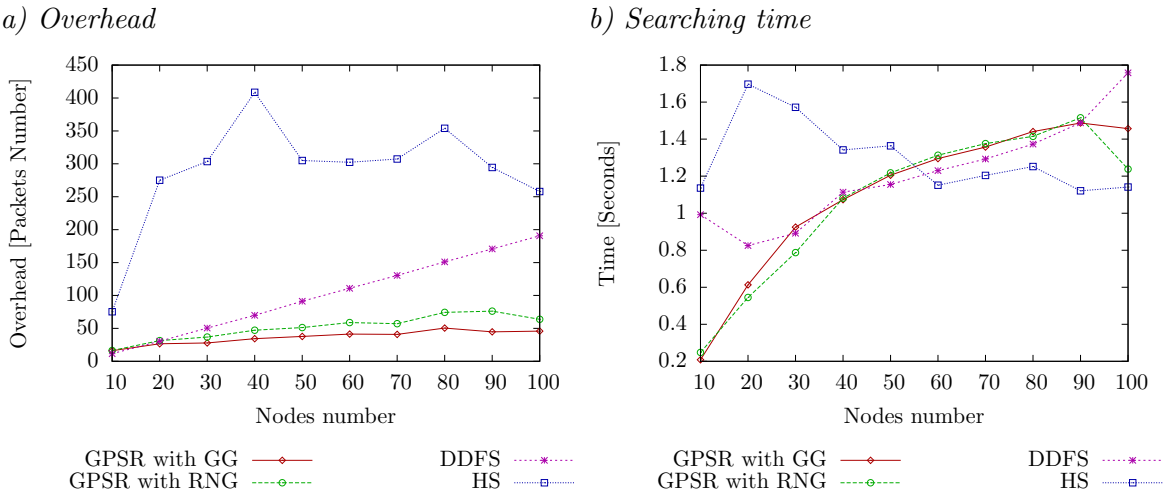


Figure 5.15: Average packet count (overhead) (*a*) and average searching time (*b*) for NNS procedure with different base algorithms

Therefore, the GPSR algorithm with the GG planarization was selected as the base for the NNS algorithm. Moreover, GPSR with GG showed also the best results for the networks with a complex topology like the topology shown in figure 5.14 on page 77, where a big gap exists between the originator node and the search destination point. The results shown here are already published in [TSA11].

5.6.7 NNS Implementation in Click

The Click router graph part responsible for the NNS procedure is shown in figure 5.16. The packets related to NNS and the mission processing mechanism come from the network to the *GPSRClassifier* element. This element sorts the incoming packets into three classes: the `BEACON` packets, the mission request and reply packets and the `GPSR` related packets.

The `BEACON` packets are collected by the *NeighborTracker* element, which maintains and updates the neighbor table and provides access to this table for other elements via

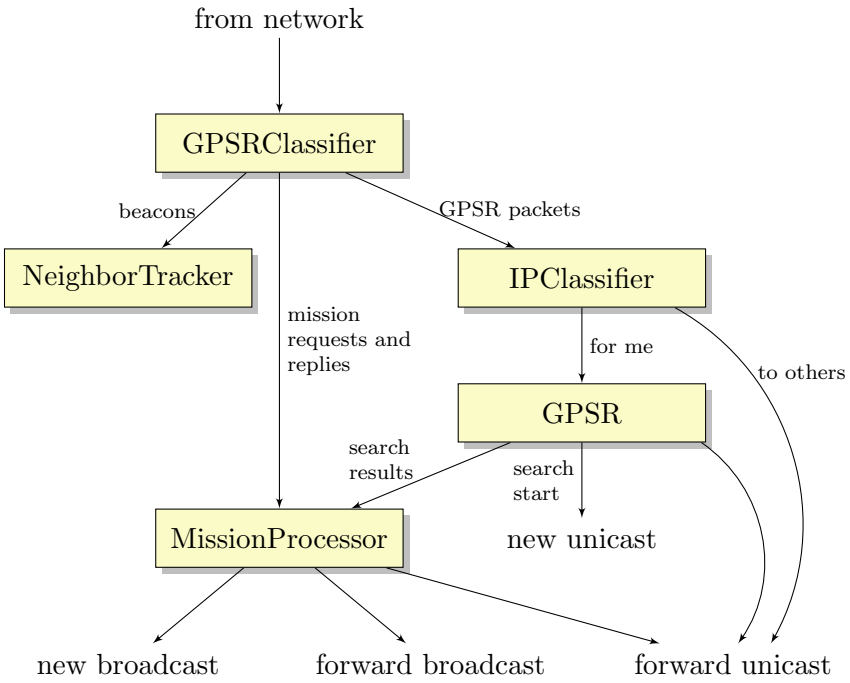


Figure 5.16: Click router graph for the NNS implementation

the method interface. Each entry in the neighbor table contains the following fields: the IP address of the neighbor, the geographical coordinates in form of latitude and longitude values and the timestamp of the last entry update. The table is cleaned within an interval equal to two beaconing intervals, so that one `BEACON` message can be lost without affecting the neighborhood information of the node.

The packets related to the mission requesting mechanism are forwarded to the *MissionProcessor* element. The details of the mission requesting subsystem are discussed below in section 5.11.

The packets belonging to the NNS mechanism proceed to the *IPClassifier* element, which separates the packets destined to the current node. After that, the packets for other nodes are redirected to the AODV routing system and are handled as normal transit IP packets. The packets, whose destination is the current node, are forwarded to the *GPSR* element.

GPSR is the element, which implements the GPSR algorithm for the NNS procedure. On the trigger event generated by the problem detection mechanism, the *GPSR* element initiates the search procedure. The information needed for the GPSR algorithm is provided by the *NeighborTracker* (list of neighbor nodes with their positions) and the *GPSReceiver* (coordinates of the node) elements as well as the search destination position provided by the trigger event itself.

The initially generated `GPSR` packet is extended with position information of the current node (s. section 5.5) and forwarded to the network. In the figure, this way is depicted

as *new unicast*.

If the *GPSR* element receives a packet, but the algorithm does not terminate in this node, the packet will be forwarded to the next hop node according to the GPSR algorithm over the *forward unicast* way. On this way, the TTL value of the IP packet is decreased before the packet goes into the network, and the IP header still contains the geographical position of the originator node.

If the search procedure is finished in the current node, the packet containing the list of the retrieved nearest nodes will be forwarded to the *MissionProcessor* element producing a mission request (s. section 5.11.1) if the list does not contain the nodes, which cover the search destination point with their own communication range. Remember, if such nodes were found, a failure of the destination node is assumed and no mission request should be generated.

5.7 Signaling and Communication Link (SCL)

Before the alternative placement approach for the situation with available knowledge about the network topology will be discussed, it is necessary to describe the architecture supporting the global knowledge exchange. The SCL [SPTK12] infrastructure was developed by several members of GS Mobicom in order to provide a unified interface between different software components designed within the graduate school.

The main idea behind SCL was to design a lightweight message-based inter-component communication infrastructure. Moreover, SCL is aimed to support the independence between three software dimensions: data serialization, message transport and topology definition.

The well known systems like the Remote Procedure Call (RPC) [Nel81] and the Common Object Request Broker Architecture (CORBA) [Gro99] create a strong coupling between these three dimensions. Also these systems force the developer to use a certain paradigm of software development: procedural programming in RPC and object-oriented programming in CORBA. In opposite to the systems, SCL gives to the developer a freedom to choose the paradigm as well as the format of used messages.

The following two messaging architectures are very popular in embedded and robotic software field during the last years: the Lightweight Communications and Marshaling (LCM) [HOM10] library, and the Robot Operating System (ROS) [QCG+09], which is not an operating system as such but a message-based middleware. Both systems provide fixed built-in data serialization mechanisms and support only the publish-subscribe communication pattern. In contrast to LCM and ROS, SCL supports any arbitrary data serialization mechanism. Along with the publish-subscribe pattern, SCL provides also the request-reply communication pattern. The support of both communication

patterns is based on the ZeroMQ (MQ stands for Message Queuing) [zer12] library, which is used as base for the communications behind SCL.

5.7.1 Design and Structure

SCL is designed as easy to use software framework providing a simple but powerful API for developers without the requirement to follow a specified programming paradigm. A system designed with SCL is not restricted to how its components are interconnected with each other, and can use different communication patterns. The SCL framework supports different programming languages and different platforms. For the purposes of the GS members, SCL implements interfaces in C, C++ and Python. SCL is based completely on open source libraries and is also the open source itself.

As mentioned before, SCL separates three software dimensions which are mapped onto three orthogonal functional blocks:

- The **topology definition** block defines the interconnections between system components together with the used communication patterns.
- The **data serialization** block specifies the format in which messages are transported within the system.
- The **message transport** block describes the transportation details, how messages can be delivered from component A to component B.

The system topology is defined in a system configuration file using the YAML (a recursive acronym for "YAML Ain't Markup Language") [yam12] format. An example of the YAML configuration file is shown in figure 5.17. In the example three system components are defined: A, B and C in the section *components*. Each component has one or more gates of one of the following types: publish (*pub*), subscribe (*sub*), request (*req*) and reply (*rep*). In the section *connections*, the system topology is defined via a list of connected gates. In the example, gate $a1$ of component A is connected to gate $b1$ of component B using the request-reply communication pattern, and gates $c1$ and $b2$ are connected via the publish-subscribe communication model. Note that only the complementary gates can be connected: *pub* with *sub* and *req* with *rep*. However, to one *pub* gate multiple *sub* gates can be connected. The same is valid for *rep* and *req* gates respectively.

As mentioned before, SCL can use any arbitrary data serialization mechanism: simple text format, any markup text format or any binary format. In the graduate school, the Google Protocol Buffers (protobuf) [pro12] format is used since it provides an effective cross-platform serialization and deserialization mechanism for the messages of any arbitrary structure in many different programming languages.

```
components:
   - name: A
     gates:
     - a1: req

   - name: B
     gates:
     - b1: rep
     - b2: pub

   - name: C
     gates:
     - c1: sub

connections:
   - ["A.a1", "B.b1"]
   - ["C.c1", "B.b2"]
```

Figure 5.17: Example of the simple YAML configuration file for SCL

For message transport, the ZeroMQ library is used. ZeroMQ provides a reliable message transportation and queuing mechanism, which supports the following kinds of communication: local in-process communication (inter-thread), local inter-process communication, unicast TCP and multicast PGM (Pragmatic General Multicast) communications. Within SCL, the inter-process communication is used, since the system components are typically separate programs (or processes in terms of operating system).

5.7.2 Data Access Module

The system providing global knowledge about the network topology needs a storage, which is accessible for all components of the system. Since the system components are interconnected via the SCL interface, the data storage should be also accessible over this interface. For this purpose, the Data Access Module (DAM) was developed. DAM provides data conversion from the internal serialized data format being used within SCL (protobuf) to SQL queries and serialization of the SQL query results into the internal data format. DAM has an extensible architecture, and the support of new data structures and queries can be added via the mechanism of plugins. The DAM structure is shown in figure 5.18.

From the side of SCL interfaces, DAM provides two gates: the *rep* gate for the request-reply communication and the *pub* gate for the publish-subscribe communication. The request-reply communication is mostly used by the system components and can be

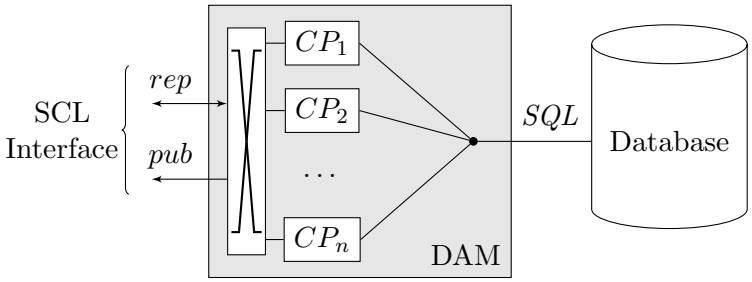


Figure 5.18: Data Access Module (DAM)

also seen as a client-server communication, where DAM works as server. The publish-subscribe communication is typically used as notification mechanism so that different system components can be notified when data will be stored or changed in database.

It is required that all ingoing messages have the first field indicating the message type. This field should be defined as a required unsigned integer field with ordinal number one in protobuf message description as follows:

```
required uint32 type = 1;
```

where *type* is the name of the field (can be set arbitrarily) and *1* is the ordinal number of the field (the field must be first). Depending on the message type, the messages are routed internally to the corresponding plugin called *Command Processor* (*CP*). Each CP is responsible for one or more message types and designed as a separate dynamically loadable shared library. So, the plugins, or CPs, make the mapping between message structure and database tables. Additionally, CPs can initiate a notification message which will be published via the *pub* gate of DAM. Thereby, DAM works as a plugins manager and provides the database connection and the SCL interface to the other system components.

DAM is also used by the network partitioning recovery system in the scenario with available global knowledge. In this case, the system gets access to a list of localized (or known) network nodes, which is stored in the database in this case, and also stores the mission requests calculated using the Voronoi diagram approach into the database for the mission planning subsystem. Note that the nodes localization is not an objective of this work and therefore is not considered in this work. It is only supposed that coordinates of the network nodes are known and stored in a database table before the moment the placement calculation is started.

5.8 Extended Voronoi Diagram

As described in chapter 4, an approach based on Voronoi diagram computation is used by the network partitioning recovery system for calculation of the placement position for additional nodes if global knowledge about the network topology is available for the

system. Like in the previous case (without available global knowledge), the task is to find two nodes with minimal distance from different network partitions. In this case, the task has a purely geometrical solution which is based on the known node positions.

As input, this approach needs a list of the nodes belonging to all network partitions with a given reference communication radius (s. unit graph model in section 4.5). At the initial phase, the system determines the connected components of the network graph which are corresponding to the network partitions. During this process, each vertex of the graph is marked with the number of the connected component which the vertex belongs to. After that, the system computes a Voronoi diagram on these vertexes in order to find the partition borders and optimal node placement positions.

5.8.1 Voronoi Diagrams

Here, the definition of the Voronoi diagrams will be given more precisely (based on definition 2.1 from [AK00]). Let S be a set of $n \geq 3$ points in a plane. For points $p = (x_p, y_p)$ and $q = (x_q, y_q)$ let $d(p,q) = \sqrt{(x_p - x_q)^2 + (y_p - y_q)^2}$ denote their Euclidean distance and $\overline{pq}$ is the line segment from p to q.

Then for $p, q \subset S$ let

$$B(p,q) = \{x | d(p,x) = d(q,x)\}$$

be the *bisector* of p and q. $B(p,q)$ is the perpendicular line through the center of the line segment $\overline{pq}$. It separates the half plane

$$D(p,q) = \{x | d(p,x) < d(q,x)\}$$

containing p from the half plane $D(q,p)$ containing q.

The intersection of such half planes for the point p with respect to all other points from S

$$VR(p,S) = \bigcap_{q \in S, p \neq q} D(p,q)$$

builds the *Voronoi region* of p. Finally, the Voronoi diagram is defined by

$$V(S) = \bigcup_{p,q \in S, p \neq q} \overline{VR(p,S)} \cap \overline{VR(q,S)}.$$

Note that the Voronoi diagram can consist of both closed and opened convex regions (see fig. 5.19).

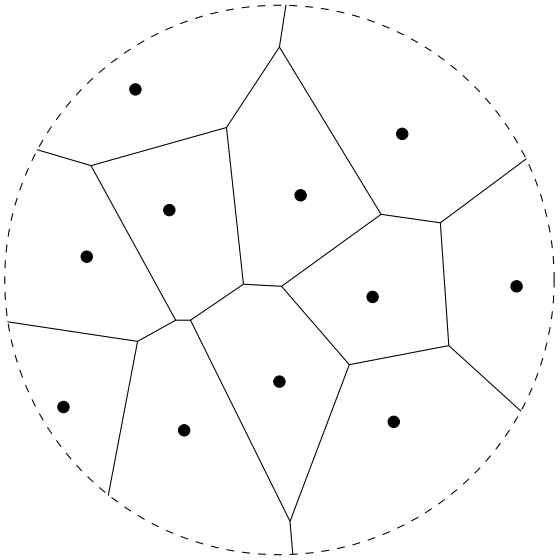


Figure 5.19: A Voronoi diagram of 11 points in the Euclidean plane (from [AK00])

5.8.2 Construction Algorithms

There are several algorithms to calculate the Voronoi diagram which are listed in the following table 5.3. It can be shown (see [AK00]) that the lower bound of calculation time complexity is $O(n \log n)$. There were some researches of algorithms optimization which

Table 5.3: Algorithms for the calculation of Voronoi diagrams

Algorithm	**Complexity**	**Source**
Incremental construction	$O(n^2)$	[GS78]
Divide-and-conquer technique	$O(n \log n)$	[SH75]
Sweep line algorithm (Fortune's algorithm)	$O(n \log n)$	[For87]
Lifting to 3-space	$O(n \log n)$	[AK00]

showed that in case of well distributed point sets, the computation time complexity can be decreased up to linear $O(n)$ by numeric computation (see [OM84, SI92]). However, it is impossible to rearrange the points' distribution in this scenario since each point corresponds to the position of a real physical network node, which can be placed arbitrarily.

Because all of the bottom free algorithms (the divide-and-conquer technique, the Fortune's algorithm and the lifting to 3-space algorithm) have the same complexity of $O(n \log n)$, which also equal to the theoretical lower bound complexity of Voronoi diagram computation, the algorithm selection criteria is based on the existence of a simple open source implementation. Such an implementation exists for the Fortune's algorithm [FO05].

5.8.3 Algorithm Extension

The Fortune's algorithm only generates a Voronoi diagram based on a given list of points. So the algorithm needs to be extended to be able to distinguish the Voronoi edges which separate the nodes belonging to different partitions. Moreover, optimal points on these edges should be also calculated by the extension.

The extension requires that the connected components of the network graph should be found in advance and the vertexes are marked with the number corresponding to the vertex's partition. The connected component detection is a well known problem in graph theory and can be solved using breadth-first search or depth-first search graph traversal techniques together with coloring of graph vertexes. The complexity of such an algorithm is linear against the number of graph edges and vertexes: $O(|E| + |V|)$, where $|E|$ is the number of edges and $|V|$ is the number of vertexes.

The extension does not add any complexity to the Fortune's algorithm since it uses already calculated distances which are needed also for the algorithm itself. If the current Voronoi edge is built by two nodes from different connected components of the graph, the edge is marked as border edge. Additionally, if the line connecting these two nodes intersects the edge, the intersection point is denoted for the edge with an integer number greater 0. The number indicates how many additional nodes are needed to reconnect the network over this line according to the given reference communication radius. For single node placement, this point is also the placement point and for multiple node placement, the additional nodes should be uniformly distributed along this line.

As a result, the extended Voronoi diagram calculation algorithm returns a list of border edges with specified optimal placement points on these edges. An example is illustrated in figure 5.20. The border edges are shown as bold black lines, and the points marked with 1 are the optimal placement positions. The "1" indicates that it is enough to place only one additional node into the marked position to reconnect the network partitions.

5.8.4 Implementation Details

The system component responsible for the extended Voronoi diagram calculation is implemented as a separate program using the SCL framework for communication with other parts of the system (over DAM). The topology data is prepared by a localization system and stored in a database.

When network partitioning recovery becomes necessary, and the recovery process is triggered by the system, the program requests the list of the localized network nodes. Using the geographical coordinates of the nodes from the list together with the given reference communication radius, the program builds the network topology graph and detects the connected components of it. During this process, the nodes (vertexes of the graph) are marked with one integer number – partition identifier, which corresponds

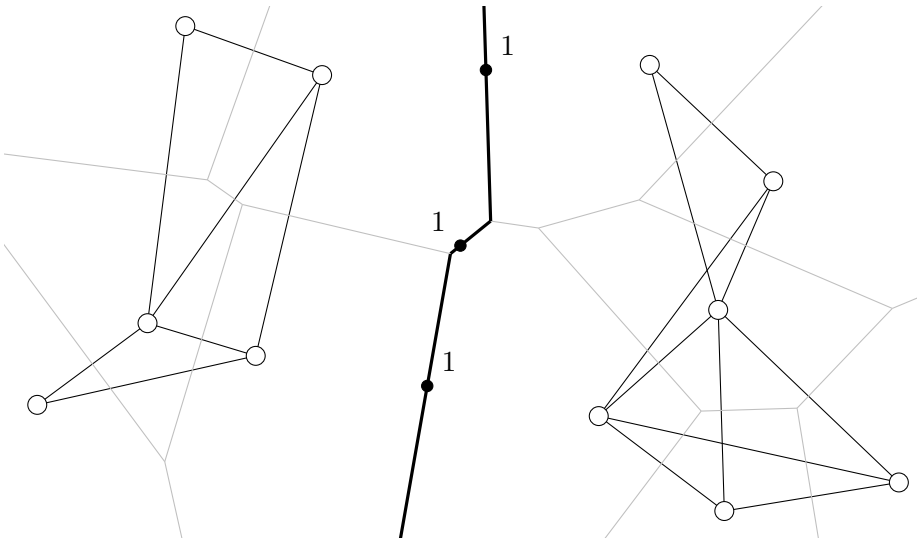


Figure 5.20: Extended Voronoi diagram with specified node placement positions

to partition (connected component of the graph) of the node. So, all nodes in one partition get equal partition identifiers.

The nodes list extended with partition information is sent to the Voronoi diagram calculation algorithm, which calculates the list of border edges as described previously. This list is stored into the database in format described in section 5.11.2 and becomes available for the mission planning system.

5.9 Nodes Placement Calculation

The calculation of the nodes placement positions differs for various scenarios with and without available global knowledge about the network topology. If the global knowledge is not available, the distance between partition border nodes is a priory unknown, and, as consequence, it is also not known how many additional nodes are needed to reconnect the partitions. In case of the centralized calculation of Voronoi diagram, the distance is known and the placement positions can be calculated in advance.

Thus, without the global knowledge available, the placement of additional nodes should have a consecutive character. Let's suppose that one nearest node was found in position A, the reference communication radius is R_{ref} and the last known position of the node, which the communication should be reestablished with, is D. Note that the positions A and D are given by their geographical coordinates. Also some guard interval Δd should be defined, which guarantees that the additional node will not be disconnected from the its own network partition during the placement process. Three following situations illustrated in figure 5.21 are possible:

- The distance between A and D does not exceed $(R_{ref} - \Delta d)$ (position D_1). In this case, a single node failure (failure of the destination node) is assumed since at least one node within the communication radius of the destination node was found, and the communication is theoretically possible.

- The distance between A and D lies between $(R_{ref} - \Delta d)$ and R_{ref} (position D_2). In this case, only one additional node is needed to reconnect the partitions. The placement position can be calculated here as the middle of the line connecting A and D.

- The distance between A and D is greater then R_{ref} (position D_3). Here it may be that more than one additional node is needed.

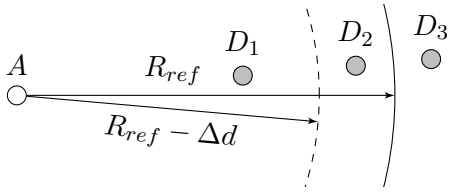


Figure 5.21: Possible last known positions of the communication partner node

For the last case, the following placement strategy can be applied. The vector $\overrightarrow{AD}$ determines the placement direction. The first additional node is placed in this direction on the distance of $(R_{ref} - \Delta d)$ from point A. If the communication with the destination node is possible again, the mission is done. Otherwise, the second additional node is placed in the same direction on the distance of $(R_{ref} - \Delta d)$ from the last placed additional node and so on until the communication with the destination node is possible or no more additional nodes are available.

In the scenario with global knowledge available, the positions of the nearest border nodes for both partitions are known and are A and B. Then, d is the distance between A and B. The number of additional nodes can be calculated as $n = Int[\frac{d}{R_{ref} - \Delta d}] - 1$, where operator $Int[x]$ returns the next closest integer for x. Then, the distances between the border nodes and the additional nodes are equal to $\frac{d}{n+1}$. For example, if the distance between the border nodes is $d = 750$ m, the reference communication radius is $R_{ref} = 150$ m, and the guard interval is $\Delta d = 15$ m, then the needed number of additional nodes is $n = 6$ and the distances between nodes are 125 m each. Note that the maximal distance between border nodes, which can be covered by 6 additional nodes, is about 810 m in this example.

However, if the network has split into more than two partitions, then according to the presented algorithms, the partitions will be reconnected pairwise. It might not always be efficient, since it may be possible to reconnect more than two partitions using only one placement. Moreover, if more than one placement are needed, it is necessary to decide which placement should be done at first, especially if the available number of additional nodes is not sufficient.

5.10 Comparison of the Placement Scenarios

Let's compare the results of the both node placement algorithms - with and without available global knowledge - for identical scenarios. The scenario is illustrated in figure 5.22. Initially, both groups of nodes were connected via nodes C and E, and a communication between nodes A and B was established. The group on the right side had the position illustrated by the node group drawn with dotted lines, where E_0 is initial position of node E, and B_0 is initial position of node B. After that, the right group of nodes has moved to the right side and the connection between the node groups was broken.

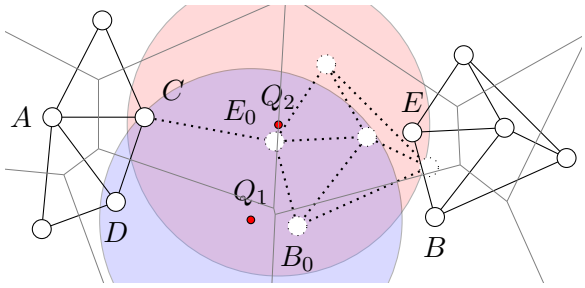


Figure 5.22: Comparison of the node placement strategies with and without available global knowledge

In the first case, the scenario without available global knowledge is considered. For the left group of nodes, the last known position of node B is B_0. So, according to the rules defined in the previous section, the placement position for the additional node will be estimated as Q_1. In this case, node D will be found by the Nearest Nodes Search algorithm as the nearest node to B_0. The communication range of the additional node is depicted with a filled circle with center in point Q_1. It is visible that the right group is out of the coverage of the additional node and it means that another additional node is necessary to reconnect the partitions.

In the second case, the global knowledge about the network topology is available for the system. In this case, the extended Voronoi diagram can be calculated for the network, and according to the algorithm, the placement position for the additional node will be estimated as Q_2. Again, the communication range of the additional node is depicted with a filled circle, but with center in point Q_2. Now, it is visible that node E is within the communication range of the additional node, and only one additional node is needed to reconnect the network partitions.

It is obvious that the node placement position can be calculated more accurately if more information about the network topology is available for the system. However, if no global knowledge is available, the first approach still can provide acceptable and useful results, in the same time the Voronoi diagram based approach cannot be applied in this case at all.

5.11 Mission Requesting System

After the placement positions for additional nodes were estimated using NNS or Voronoi diagram approaches, the system generates a mission request to the system controlling the mobile nodes – the mission planning system. The main tasks of the mission planning system are the reception of mission requests, the missions allocation between the mobile nodes, and the route planning with respect to available mobile nodes and their resources like power or communication range. The mission planning system is a part of a separate PhD topic within the graduate school and is not described here in detail. The mission planning system can also be present in two variants: distributed and centralized. Depending on the realization variant, the mission planning system provides two different interfaces.

5.11.1 Mission Requesting without Available Global Knowledge

In the use case, where no global knowledge about the network topology is available for the system, the interface of the mission planning system is based on simply flooding mission requests. Each mission request (`MREQ`) has a form of a broadcast UDP packet. The mission requests are broadcast because the address of the mission planning system is unknown for the network nodes, and the route cannot be found by a routing mechanism present in the network. Since a network partitioning causes typically more than one communication problem, which are detected by the network partitioning recovery system, more mission requests with similar goals will be generated, and an unacknowledged transport can be used for the mission requests. Additionally, the unacknowledged transport allows to avoid communication overhead caused by the acknowledgements and by possible packet retransmissions. The mission request packet format used with IPv4 is shown in figure 5.23.

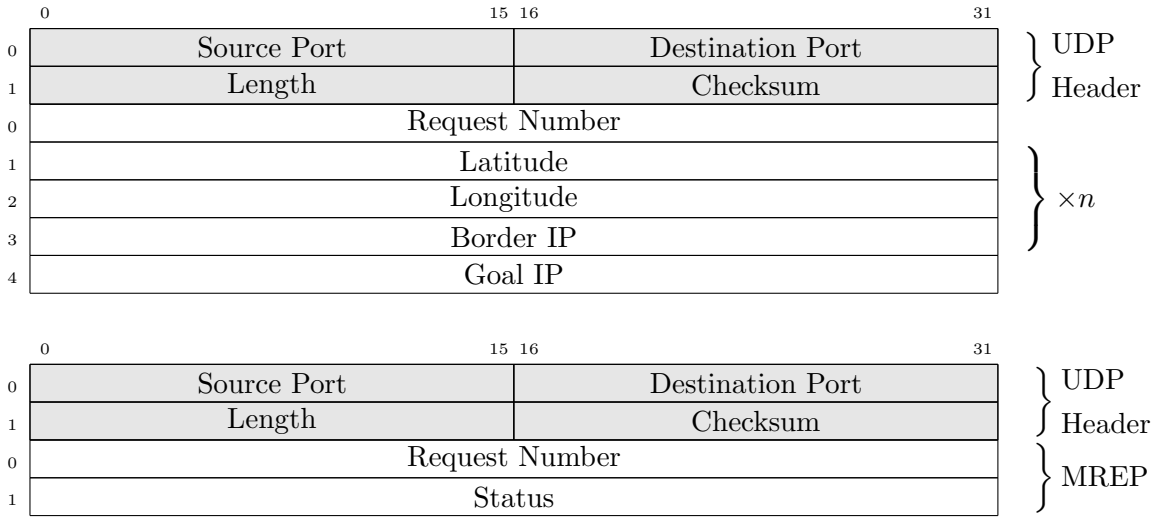


Figure 5.23: Mission request (`MREQ`) and mission reply (`MREP`) messages format

The UDP source and destination port numbers are set to 942, and the IP packet encapsulating the mission request has a broadcast destination address. The `Request Number` 32 bits field contains the sequence number of the mission request. Together with the source address of the packet, it builds a unique mission identifier. The `Latitude` and `Longitude` fields contain the geographical coordinates of the placement point, and the IP address of the corresponding border node found by the NNS procedure is stored in the `Border IP` field. The last three fields are repeated n times according to the number of nodes found by NNS. The last field `Goal IP` contains the IP address of the goal node, which the NNS procedure was initiated for.

Optionally, the mission planning system can generate a mission reply message addressing the node, which generated the corresponding mission request. The mission reply contains the mission number (`Request Number`) and the request status (`Status`) – an unsigned 32 bits integer value, where 0 indicates a successfully accepted mission and other values are reserved for error reporting.

Within the Click-based implementation, the mission request functionality is provided by the *MissionProcessor* element (s. figure 5.16 on page 81). The results of NNS are pushed to the *MissionProcessor* element, which generates a broadcast UDP packet containing the mission request as response on this event. The mission request is forwarded in the *new broadcast* direction. If the *MissionProcessor* element receives a mission request from the network, it rebroadcasts the request again with the constraint, that the mission requests, which were already seen by the node, will not be retransmitted again. The retransmitted mission request packet follows the *forward broadcast* way, where its TTL value is decreased.

If the mission planning system generates a mission reply, it is forwarded as unicast packet using the ordinary routing mechanism. The mission replies are also passed through the *MissionProcessor* element in order to avoid the propagation of possible following mission requests with the same goal node IP or with similar destination point coordinates.

5.11.2 Mission Requesting with Available Global Knowledge

In case the global knowledge about the network topology is available, the placement calculation and the mission requesting are done centralized. The mission requests are stored in the database using DAM, so that it can be also accessible for the mission planning system over the same DAM interface. The mission requests stored in the database table contain the following fields:

- `id` – a unique mission identifier. Over this identifier, the mission is accessible for updates and changes.
- `type` – the mission type. The following types are available:

cov The type specifies the task of the network coverage improvement. The mobile nodes are asked to be placed as additional wireless access points or base stations. The type is reserved for future use.

qos The goal of such missions is QoS improvement in some parts of the network. This type is also reserved for future use.

con The missions of this type are aimed to reconnect the network partitioning.

loc This type is used to specify the missions for the nodes localization process, if the localization is done using the mobile nodes.

- `priority` – specifies the mission priority. The field is reserved for future use and has the default value 1.

- `status` – contains the actual mission status. The following statuses are available: *requested*, *rejected*, *active*, *waiting* (the mission is accepted, but it needs to wait for some resources), *pause* (pause in an active mission, e.g. caused by a new mission request with a higher priority) and *done*.

- `points` – a list of points describing the mission. The points are separated in the list with semicolons and specified in the following format:
  `<latitude>,<longitude>,<priority>`
  The *priority* parameter here characterizes the point, if the mission request contains a list of pre-calculated suboptimal placement positions. For example, for mission requests generated during the node placement calculation by the Voronoi diagram approach, the *priority* corresponds to the number of additional nodes, needed to reconnect network partitions over this point. The more mobile nodes are needed, the lower the priority is set for the point.

- `timestamp` – contains the Unix timestamp of the last mission update.

In DAM, a separate plugin is responsible for accessing the mission table. The plugin utilizes both DAM interfaces (*rep* and *pub*, s. section 5.7.2). The mission requests and updates are processed over the *rep* interface using the client-server communication model and are reflected to all subscribers, e.g. the mission planning system, over the *pub* interface. So, the system components can be notified of each mission update by subscribing to these events.

5.12 Aggregation

As already mentioned in the previous chapter, if only two isolated partitions were built in the network, a number of communication problems can be caused by it. Even on the scale of one node, the network partitioning can lead to several problems being detected,

if the node has several connections with nodes from the separated partition. If each problem detection initiates a separate NNS procedure, the network becomes quickly overloaded and all communications may be broken. Of course, this problem can take place only in the decentralized scenario without available global knowledge about the network topology. Further in this section, the distributed scenario is implied.

From the point of view of one isolated network partition, the following situations are possible: a node from the current partition has several links to a node from the other partition, a node from the current partition has several links to several nodes from the other partition, and more generic, a node from the current partition has several links to several nodes from different partitions (if the network splits into more than two parts).

On the other hand, it may be possible that the placement of only one node can reconnect more than two partitions. So, it can be seen that several aggregation levels are possible. The aggregation layers are shown on figure 5.24. On the first layer (I),

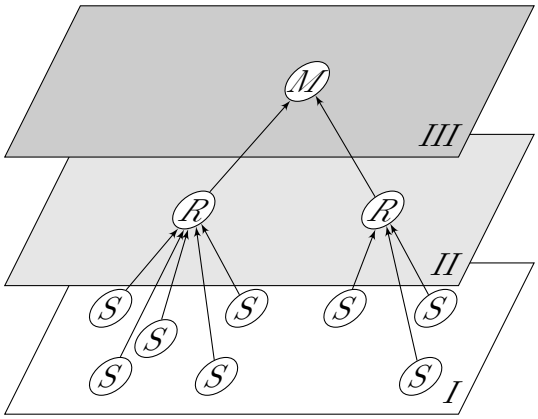


Figure 5.24: Three levels of aggregation: I – individual searches; II – aggregated mission requests; III – aggregated missions.

individually detected problems can be aggregated to one NNS procedure request. For the aggregation, the following two parameters can be used:

- Equal IP destination addresses of the node's links. Such a kind of aggregation succeeds automatically in the problem detector, since it does not distinguish different connections to the same communication partner node.

- Close last known positions of the communication partners. If the node detects communication problems for two communication partner nodes with their last known positions closer than $R_{ref} - \Delta d$ (R_{ref} is the reference communication radius and Δd is the guard interval, see also section 5.9), then both nodes belong to one partition with a high probability. The nodes are also one-hop neighbors, and if the connection with one of them will be reestablished, the communication with the other should be possible, too.

This kind of aggregation forms the lowest layer of aggregation and is carried out by individual nodes.

Based on the same principles, the mission requests generated by different nodes can be aggregated on the second layer (*II*), in order to reduce the propagation of the flooding mission requests. So, for example, if a node receives a mission request with the same goal IP address or similar placement positions as by one of previous requests, it does not rebroadcast the request.

On the third layer (*III*), the received mission requests can be aggregated to one mission by the mission planning system. On this level, the aggregation depends also on the available resources of mobile nodes. Theoretically, it is also possible, that the missions of different types can be combined: e.g. QoS improvement on the partition border with reconnection of partitions.

5.13 Summary

In this chapter, the main building blocks of the network partitioning recovery system were presented with their functional description and implementation details. Since the proposed system has two different usage scenarios (with and without available global knowledge), the parts of the system related to the both scenarios are based also on different software architectures. The implementation of the system components for the distributed approach uses the Click Modular Router software framework. Behind the system components, which are used if global knowledge about the network topology is available, lies the SCL architecture for system components communication. However, some parts of the Click-based implementation can be utilized to support the centralized approach. In this case, the Click-based implementation is still useful for routing and problem detection mechanisms.

The Click-based implementation is designed to work in user space mode on a Linux platform. Usage of the user space mode provides more flexibility by the software design, since all native Linux libraries are available here. Furthermore, program debugging in user space is easier since the system is protected against the errors in the program, whereas an error in kernel space program could lead to crash the whole system. However, the usage of the user space program has its own drawbacks: the programs are working in general more slowly in user space than in the kernel space, and an intermediate software layer is needed to provide the access to the network protocol stack, which resides in kernel space.

The following chapter provides the description of simulation and validation scenarios for the network partitioning recovery system. The instruments, used for simulation and emulation will be also discussed there.

6 System Evaluation

This chapter deals with the evaluation of the network partitioning recovery system. The system evaluation is done in several phases: simulation of different parts of the system and of the whole algorithm in a network simulator, system emulation on a real Linux platform within a virtual environment and tests in a demonstrator.

The first phase allows to prove the concept of the developed algorithms and to test the efficiency. Also the collaboration of the different system parts can be tested here. Usage of a network simulator also provides various mechanisms for collecting statistical data and performance evaluation. The major advantages of simulation are the ability to work with a large number of nodes and the flexibility by selecting the simulation accuracy, so that some at the moment unimportant details can be excluded from the simulation. However, the implementations used in this phase are not real and tightly related to the environment provided by the used network simulator and they also depend on the simulator architecture.

In opposite to the first phase, the second phase considers the real algorithm implementations, which can be also used within the real network. Additionally, the virtual environment used here provides the possibility to control and to monitor the whole network recovery process on the level of separate messages transmitted by the network nodes and allows to model different network scenarios. However, the number of the network nodes is limited here by the performance of the computer used for the emulation, since each node is presented by a virtual machine.

The last verification phase is aimed to test the system in the real environment with real devices. This evaluation phase is most expensive, since it needs a number of mobile devices, each of which should also be configured before it can participate the system. Here, the total number of nodes is highly limited by the available devices.

In this chapter, the presented three phases will be discussed in detail.

6.1 System Simulation

The simulation of separate system parts was widely used during the algorithm development process. The simulation allows to concentrate on the key parameters of the developed algorithms without taking into account the details which are not important

for the algorithm itself. Thereby it is possible to build the simulation on different abstraction levels.

An important thing for simulation is the selection of the suitable instruments. In the following section a short overview on the most famous network simulators will be given.

6.1.1 Network Simulators

A number of network simulators is developed and used by different researches. There is open source and proprietary simulation software, aimed to solve the tasks of a specific domain or oriented to different domains. Some simulators have the goal of a precise simulation of the physical layer and provide different signal propagation models and medium access schemes. Other simulators are message oriented and do not take into account the problems of the lower layers. There are also simulators oriented to network applications and services. However, there are network simulators of common use, which are able to cover the whole spectrum from the physical signal propagation to the network applications and services in one simulation.

Within the scope of this work, only the major open source simulators of common use were taken in to account, since the different network simulators are in general incompatible to each other but the simulation should be possible on different layers so that the simulation codes are also reusable in different simulations.

The three most famous network simulators will be discussed below: NS-2, NS-3 and OMNeT++.

6.1.1.1 NS-2

NS-2 is a discrete event simulator targeted at networking research. NS-2 provides substantial support for simulation of TCP, routing, and multicast protocols over wired and wireless (local and satellite) networks. [LAB+11] A simulation scenario is described using the Object Tcl (OTcl) language – an object-oriented extension of the Tool Command Language (Tcl). The simulation scenario contains the information about the structure of the simulated network, the used protocols and applications, the nodes' movement and others simulation parameters. For the specification of the network architecture, the basic model illustrated on figure 6.1 is used. Firstly, all the network nodes used in the simulation with their links should be declared. Each nodes link corresponds to the physical connectivity of the nodes. For the link, channel characteristics (wired, wireless, delays, etc.) and packet queuing mechanisms can be specified. For example, the following string in OTcl language

```
$ns duplex-link $n0 $n1 1Mb 10ms DropTail
```

defines a duplex wired link between nodes $n0$ and $n1$ with the bandwidth of 1 Mbit/s and the delay of 10 ms (packet propagation time) with a simple model of drop-tail

queue (if the queue is full the arriving packets will simply be dropped). The term `$ns` is a reference to the simulator object.

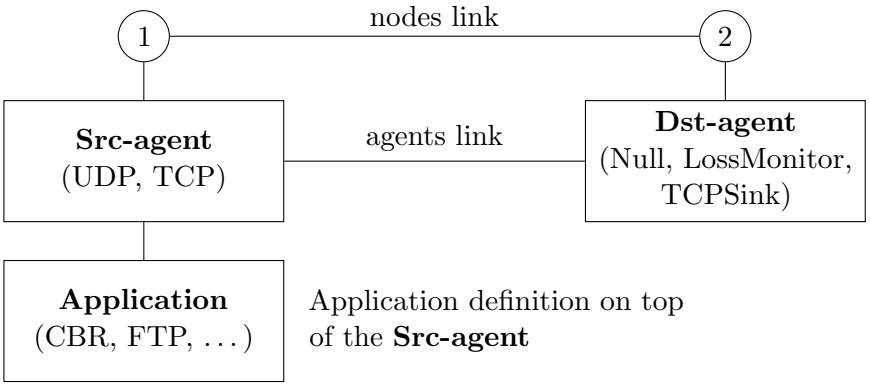


Figure 6.1: NS-2: Basic simulation model

After that the nodes and their links were defined, source (*Src-agent*) and destination (*Dst-agent*) agents should be attached to the nodes, which generate or consume traffic. The agents present a kind of protocol stack for the nodes, and the source and the destination agents should be connected to each other.

Finally, applications should be defined on top of the source agents. The applications simulate some traffic patterns which are typical for applications. For example, a Constant Bit Rate (CBR) application can simulate a multimedia stream, which consists of similar (constant) data volumes being sent periodically. In opposite to CBR, a File Transfer Protocol (FTP) application generates traffic bursts within irregular time intervals.

As mentioned before, the simulation scenario is to be defined in the OTcl language. However, OTcl only provides an interface to simulator objects (simulator itself, agents, application, channels, etc.) written in C++. In order to add a new protocol to the simulation, a C++ class with the corresponding OTcl interface should be implemented. Also, some common files of the network simulator itself are affected by the extension, and the complete simulator program must be newly compiled after each extension.

The NS-2 simulator is delivered with a simple Graphical User Interface (GUI) program NAM (from Network Animator). In fact, the NAM program can only animate the log-files written during the simulation. So, it does not allow any interaction, which could affect the simulation.

NS-2 defines its own trace file format, which is accessible by NAM. The trace file contains the description of separate simulation events. The following four types of events are predefined: r – received at destination node, $+$ – enqueued, $-$ – dequeued and d – dropped at a queue. Each event includes such information as time, source and destination addresses, packet size and type and some other additional parameters. In

the last versions of NS-2, a new trace file format is available. The new format is more flexible and allows to specify an arbitrary number of parameters in a tag-value format for each event. The supported event types are: send, receive, drop and forward.

To collect any statistical information, the trace files should be analyzed by some external tools. For example, statistical calculations can be done in a *perl* script, and the *gnuplot* program can be used for drawing some diagrams. The NS-2 distribution itself does not include any analytical tool at all.

6.1.1.2 NS-3

The second network simulator, which will be discussed here is the NS-3 simulator. NS-3 is a discrete-event network simulator for Internet systems, targeted primarily for research and educational use [HRFR06, LW+12]. NS-3 is the next generation of the Network Simulator (NS-2) and is incompatible with NS-2. The simulator was completely redesigned, so that the simulations implemented for NS-2 cannot be started in NS-3, since NS-3 does not use the OTcl interface anymore. The whole simulation scenario for NS-3 must be written in C++. Also, a *python* binding is available.

The NS-3 simulator has an extensible software core written in C++ with attention to realism due to a socket-like interface, which is used for simulation programs. The basic simulation model for NS-3 is shown in figure 6.2. Each node participating in the simulation contains one or more network interface device models (*NetDevice*), a *Protocol Stack* providing the socket-like API to *Applications*. The nodes are connected via channels (wired or wireless). The whole network topology, the structure of the individual nodes and the simulation scenario itself are defined in C++ program. Since the simulator pays attention to realism, a large number of parameters should be defined before the simulation can be started. To simplify this procedure, a number of *Helper* classes is defined for different abstractions introduced by the simulator.

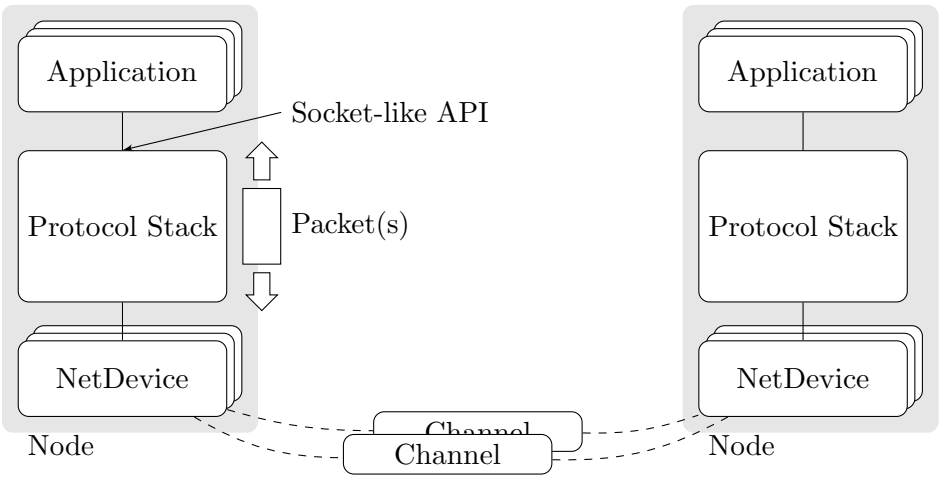


Figure 6.2: NS-3: Basic simulation model, taken from [Hen10a]

In NS-3, the following key abstractions are defined:

- **node** – network end system, basic computing device abstraction, class *Node*;
- **application** – basic abstraction for user program, activity generation, class *Application*;
- **channel** – basic communication subnetwork abstraction, media, class *Channel*;
- **net device** – basic abstraction for Network Interface Cards (NIC) and drivers, class *NetDevice*;
- **topology helpers** – interconnections between model components, address assigning, classes *Helper*, *Container*.

In order to add to NS-3 the support of a custom protocol, an application or a channel model, the extension should be implemented as a C++ class derived from one of the classes representing the key abstractions listed above.

The NS-3 simulator supports virtualization and testbeds, so that the simulation can interact with a real network. It can be achieved due to the binary compatible packets generated by the simulation program. This binary compatibility make it also possible to use standard tools for the analysis of network traffic like *tcpdump* [FHR+11] or *Wireshark* [C+12]. Additionally to packet tracing, NS-3 offers flexible logging mechanisms producing any additional statistical information being needed.

NS-3 does not have its own GUI, but it can produce the trace files in NS-2 format, which can be played with NAM. For the results analysis, external tools are also needed as for NS-2. However due to the flexible logging mechanisms, NS-3 allows to collect the information for the statistical analysis in a form, which can be directly adapted to the format used by each concrete tool.

6.1.1.3 OMNeT++

The third simulator discussed in this work is OMNeT++. The developer community defines it as follows: OMNeT++ is an extensible, modular, component-based C++ simulation library and framework, primarily for building network simulators. [V+12]. OMNeT++ offers a flexible architecture, and due to a big number of extension libraries and frameworks, it can be used for simulation of different types of networks: wired and wireless communication networks, WSNs, on-chip networks, queuing networks, etc.

OMNeT++ is delivered with an Eclipse-based IDE and also provides graphical and text-based simulation runtime environments. The graphical simulation environment allows to run the simulation step-by-step or continuously. Furthermore, some simulation parameters can be changed on-line in this environment. In opposite to the graphical

environment, the text-based environment is not interactive and uses a predefined set of parameters for the simulation, but it allows to run the simulation multiple times in batch mode with possibly changing parameters. Both environments support multilevel tracing of the simulation. OMNeT++ also offers various output file formats for collecting different statistical informations:

- **.anf** output format definition and filter. The file defines, which available statistical information should be collected and how this information should be presented (table, chart, etc.).

- **.vec** vector output data. The vector data is coupled with some simulation event and presents a set of model parameters, characterizing the internal state of some simulation object, at a given point of time.

- **.sca** scalar output data. The scalar output data comprises such parameters, which can be presented by a single value for the whole simulation run (e.g. number of packets processed by a node).

- **.elog** events log-file. The events log-file contains the complete list of the simulation events which can be used for message-sequence-charts.

- **.nam** NAM trace file. The NAM trace file, compatible with the trace files of NS-2, can be produced during the simulation (supported by an extensions library).

- **.pcap** packets trace file. These trace files contains the captured packets and can be analyzed with *tcpdump* or *Wireshark* (supported by an extensions library).

The basic model of OMNeT++ uses the concept of interconnected components and is shown in figure 6.3. The components are connected via gates with links defined by a channel model. OMNeT++ allows simple and compound components (modules). Such an architecture makes it possible to build the simulation on different abstraction levels. So, a network node can be presented by a simple component as well as by a set of components, each of which implements a particular protocol or a network device. Thanks to this feature, it is possible in OMNeT++ to simulate both the behavior of a separate algorithm and the whole network providing different services.

The OMNeT++ simulator is implemented in C++ and also uses various file formats for different purposes. For each component (simple and compound), a **.ned* file should be provided. This file includes the description of the component's interface: available input and output gates, model parameters possibly with default values, graphical view (e.g. position and icon) and the structure (for the compound components). The network structure is also to be defined in a separate **.ned* file. The values of the model parameters, which should be used in simulation, are defined in an **.ini* file. Furthermore, the **.ini* file contains global simulation parameters like tracing level, simulation time limits, etc.. If a model needs to add the support of a new message format, it should be defined in a **.msg* file, from which a special C++ class is generated.

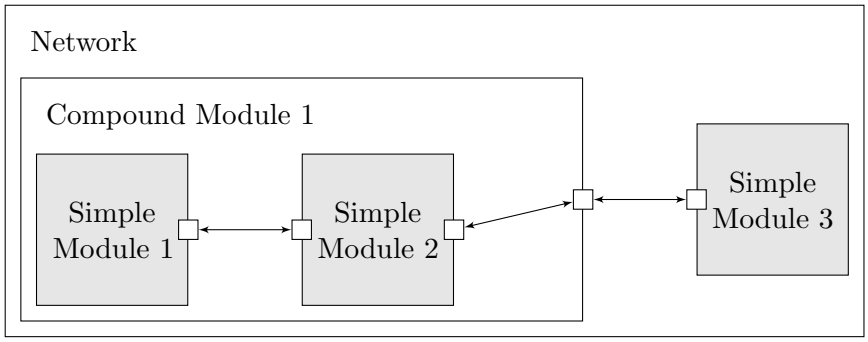


Figure 6.3: OMNeT++: Basic simulation model

As mentioned before, there is a number of extension libraries and frameworks for OMNeT++. For the simulations presented in this work, the INETMANET framework [Hor08] was used. The framework contains the models for different components of wired (ethernet) and wireless (802.11) networks: channel models, protocol implementations from different layers of the ISO/OSI model (MAC, IP, TCP, UDP, ICMP, etc.), routing protocols implementations (e.g. AODV, OLSR), movement models.

An example of a mobile host model provided by the INETMANET framework is shown in figure 6.4. In the middle, the *MobileManetRoutingHost* model is shown. The model consists of the following modules:

- *wlan* – wireless LAN adapter, an instance of the *Ieee802gNicAdhoc* module (fig. 6.4, right);
- *networkLayer* – model of IPv4 based network layer, together with ARP address resolution and ICMP error management (fig. 6.4, left);
- *manetrouting* and *routingTable* – model of the MANET routing mechanism, the routing algorithm can be set as a model parameter;
- *tcp* and *udp* – implementations of the TCP and UDP protocols;
- *tcpApp[]* and *udpApp[]* – connection points for applications;
- *pingApp* – *ping* application;
- *interfaceTable* – table containing the list of available network interfaces with their addresses;
- *mobility* – mobility model for the node;
- *notificationBoard* – internally used component for the simulation of wireless communications.

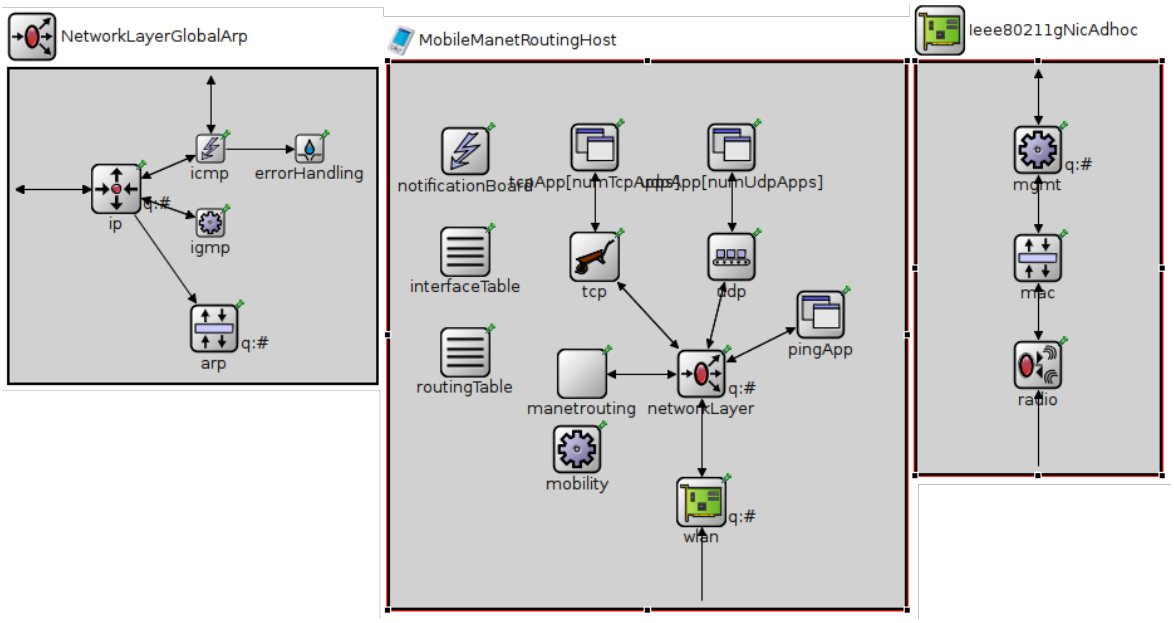


Figure 6.4: Mobile host model in OMNeT++/INETMANET

The host model shown in figure 6.4 allows a detailed simulation of a wireless node. Due to a high accuracy of the model, the simulation produces the binary correct packets, which can be captured into a *.pcap* file and analyzed with *tcpdump* or *Wireshark*. However, the high detalization grad costs high computational effort for the model calculation. For development and testing of particular network algorithms, it is typically rational to use simpler models. It is reasonable to use such a complex model only for the modeling of the complex system as a whole, in order to have the ability to consider the possible side effects caused by the interaction of different algorithms.

6.1.1.4 Evaluation

A short comparison of the network simulators discussed above is aggregated in table 6.1.

The choice of a concrete network simulator is not a simple question. There is no common answer, which simulator is better to use. The selection criteria are depending on many features: simulator use cases and simulation goals, usability (GUI, scripting languages, format of results), available protocol implementations and even the taste of developer.

For this work, the OMNeT++ simulator was selected based on the following principles: the most flexible and well structured basic simulation model allows to build simulations on different abstraction levels.

Due to extensions frameworks, all needed protocol implementations are available. The simulator provides multiple tracing formats and analytical tools. The preparation of a

simulation does not require to much programming overhead. It is easy to concentrate only on the algorithms functionality.

Table 6.1: Comparison of NS-2, NS-3 and OMNeT++

Property	**NS-2**	**NS-3**	**OMNeT++**
Source	`www.isi.edu/nsnam/ns/`	`www.nsnam.org`	`www.omnetpp.org`
Type	discrete event simulator		
Application area	wired and wireless networks, MANETs	wired and wireless networks, MANETs	wired and wireless networks, MANETs, WSNs, on-chip networks
Operating system	Linux, FreeBSD, Windows (cygwin)	Linux, Windows (cygwin)	Linux, Windows (mingw)
Basic model	nodes, channels, agents, applications	nodes, protocol stack, applications, channels, net devices	interconnected modules (simple and compound), channels
Languages: extensions	C++, OTcl	C++, Python (optional)	C++, NED
Languages: simulation scenarios	OTcl	C++, Python (optional)	NED, INI-files
Analytic tools	external	external	charts, tables, message-sequence-charts
Advantages	well documented, more protocols are implemented, large community	well structured, multiple tracing levels, compatible with PCAP format	flexible architecture of simulation models, IDE, interactive simulation environment, extension libraries and frameworks
Disadvantages	any extensions need changes in source code, implementations in two languages: C++ and OTcl	complex scenarios are hard to define, a lot of *Helper* objects needed	less protocols are implemented compared to NS-2, incomplete documented libraries

6.1.2 Simulations

In this work, the simulations implemented in OMNeT++ were used to select the base algorithm for NNS as described in section 5.6.6. For this purpose, the following modules were developed:

- *MobileManetRoutingHostEx* – extended model of a MANET host supporting AODV routing, compound module;
- *StatMobility* – extended mobility model providing the access to the node coordinates for other components (modules), plays the role of a GPS receiver;
- *NetworkLayerGlobalArpEx* – network layer model based on the *NetworkLayerGlobalArp* model from INETMANET, compound module;
- *IPn* – extended IP protocol entity supporting the transmission of geographical coordinates in the options field of IP headers;
- *NeighbourTracer* – central element for the simulation, includes all mechanisms needed for NNS: beaconing and neighbor tracking, three base search algorithms (GPSR, HS and DDFS).

The base algorithm, which is used for NNS, can be selected by parameter *searchAlgorithm* of the *NeighbourTracer* module. Also another parameter can be configured for the simulation: the algorithm specific parameters for HS and DDFS, the reference communication radius, the goal position for NNS, etc. Please note that the model does not include any problem detection mechanism, and the search procedure is initiated manually by the simulation program. This simplification allows to concentrate only on the behavior of the search algorithms.

The details of the simulation scenarios were already discussed in section 5.6.6. In the next section, the emulation of the network partitioning recovery system will be presented.

6.2 Emulation in a Testbed

The next step in the evaluation of the network partitioning recovery system is the emulation of the system in a testbed. The testbed used in this work consists of a network of virtual machines. Such a testbed setup allows, on the one side, to test the developed algorithms in a real software environment (Linux platform) and, on the other side, to keep track of the tested network as a whole. With some assumptions, it can be seen as testing the real network in some artificial environment. This structure of the testbed makes it possible to observe the network from the point of view of particular nodes as well as from the point of view of the nodes' interaction.

6.2.1 Testbed Architecture

The testbed is based on the light weight Linux-based virtual machines – the Linux Containers (LXC) [L+11]. Each LXC has its own file tree and isolated processes and resources management infrastructure. Due to resource constraints of the host PC, the actual testbed consists of eight virtual machines. All the virtual machines have the same structure and, due to this fact, the file trees used by the containers can be split into two parts: the constant part identical for all containers and the variable part containing the configuration files individual for each machine. In order to save disk space of the host PC, the constant part can be shared between all the containers. First of all, the constant part of the file tree includes applications, daemons and libraries, which stay constant during the whole life time of the virtual system. These parts of the file tree are bound to the root file system of each of the containers. The variable parts of the file tree are stored directly for each container.

Each virtual machine has a network interface device connected to a bridge device of the host PC. Over these interfaces, the machines can communicate with each other and with the host. The addresses of the interfaces (MAC and IP) are specified in configuration files of the containers.

There are several ways to interact with the containers from the outside: the console terminal, the *ssh* terminal connection and the mechanisms of inter-process communication in Linux (FIFOs and pipes). Since the network connection is not established at any time for the containers, the *ssh* connection cannot be used for the testbed. The console terminal offers an interactive environment to control the virtual machines, however all operations should be done manually. The inter-process communication offers the mechanism supporting batch operation processing, but it needs a special software being installed and running on each virtual machine. In the testbed, both communication mechanisms (the console terminal and the inter-process communication) are used.

For the purposes of the testbed, additional software should be installed on the virtual machines: software building and debugging tools, Click, a command daemon for automatization. Due to the shared part of the file tree, it is necessary to build and install the software only on one of the virtual machines, after that the software becomes automatically available on the others machines. The building and debugging software consists of the standard compiler toolchain for C/C++ and of the *gdb* debugger, so that the software used within the testbed can be compiled directly in that environment where it will be used. The command daemon together with the Linux pipes mechanism provides an interface, over which an arbitrary command from the host PC can be executed within the virtual machine. For this purpose, the command should be written in a special FIFO-file on the virtual machine. In order to automatize the work with the testbed, several scripts are implemented on the host side. The scripts allow to start or stop all the containers belonging to the testbed and also to simplify batch operations needed to be executed in each container.

An important aspect of the testbed are debugging mechanisms available for the developer. The following three mechanisms can be used with the testbed: the console output, the *gdb* debugger and *Wireshark* [C+12] or *tcpdump* [FHR+11]. Via the console output mechanism, some informational messages can be shown during the execution of the software for each individual node. Some more debug information can be obtained from the debugger, which also allows to check particular execution steps in the software. On the level of the whole network, traffic between nodes can be monitored using *Wireshark* or *tcpdump* programs running on the host platform.

6.2.2 Testbed Network Structure

As mentioned before, the testbed network consists of eight virtual machines – nodes. The MAC and IP addresses are assigned statically to each node in LXC configuration files. For the convenience of use, the Click configuration scripts describing the behavior of the nodes are split into two parts: the first part contains the information different for each node (node addresses and initial coordinates) and the second part is common for all nodes.

In order to support the artificial network environment for the testbed network, additional Click elements and a host program were implemented. The host program called *NetViewer* is a front-end for the testbed. It provides a simple GUI which shows the network topology and the positions of the nodes in real time (s. fig. 6.5). Additionally, *NetViewer* maintains the list of the network nodes with their coordinates which is accessible from the Click script running on each node. The access to this list is provided by the *Environment* element. Over this element, the nodes list can be requested and is updated each time the node changes its position. The information from the list is also used by the *GeoFilter* element, which filters the packets based on the distance calculated between the packet originator and the current node. If the distance exceeds the value of the reference communication radius, the corresponding packet is dropped, so the network topology can be maintained.

Additionally to the location functionality, the *FakeGPSReceiver* element provides a simple goto-and-stop movement model. The goal position for the movement can be specified either via the handle interface manually, or via the method interface automatically by other elements. The first interface is used if it is necessary to change the node's position from the outside, and the second interface is used by the node emulating an additional freely movable node in order to place the node into the calculated position.

There are two types of node: the normal nodes and one additional node. The nodes of both types support AODV routing. The normal nodes also include the mechanisms for NNS (based on GPSR) and for the mission request generation and forwarding. The additional node does not support NNS but has the server part of the mission requesting system implemented. So, the whole network partitioning recovery scenario

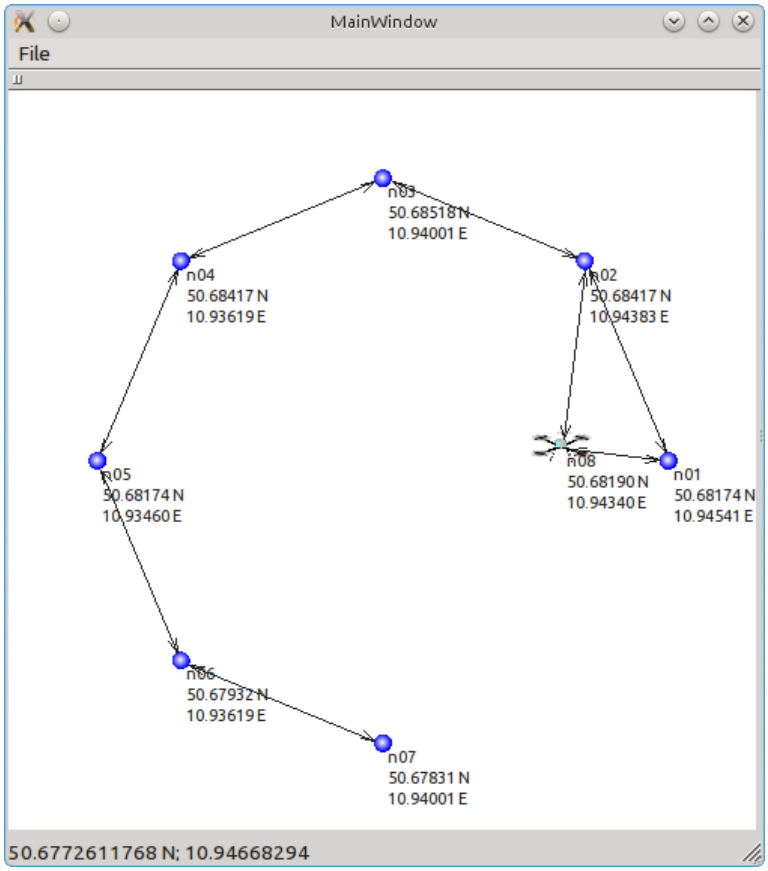


Figure 6.5: NetViewer GUI

can be emulated within the testbed. A typical test scenario is depicted in figure 6.6 and consists of the following steps:

1. One node (A) pings one other node (B) using a multi-hop communication with AODV routing.

2. One node in the chain between A and B is disabled (manually, simulated failure).

3. Node A detects the problem and tries to find an alternative route (a priori, the alternative route does not exists).

4. Node A initiates the NNS procedure.

5. Node A calculates the goal position base on the NNS results and generates a broadcast mission request.

6. The mission request propagates trough the network until it reaches the additional node QC.

7. Node QC accepts the mission request and moves to the specified position.

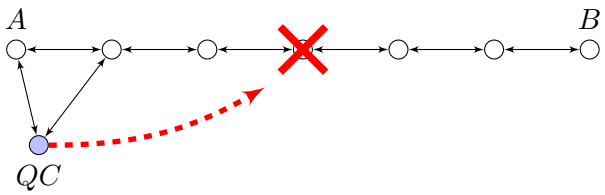


Figure 6.6: Typical scenario for the testbed

8. The ping between A and B works again and the network recovery is done.

During the test execution, the network topology information is collected by the *NetViewer* via the Linux FIFOs using the *Environment* element, and the network structure can be observed in GUI in real time.

The goal of the testbed implementation was the debugging and the proof of concept for the network partitioning recovery process. The experiments have also shown, that the main influence on the recovery time (the time interval between partitioning and the recovered connection) is given by the following two factors:

- The timing parameters of the routing – the time during which the network tries to find a new route to the destination;
- The maximal speed of the additional node and the distance between its current position and the destination.

In opposite to this, the time taken by the NNS procedure and by the mission request forwarding are insignificantly short. Typical values from emulations are 0.011 and 0.005 seconds respectively. For the recommended AODV implementation, which uses the expanding ring search algorithm (section 6.4 of RFC-3561 [PRD03]) for route discovery, with default parameters (section 10 of RFC), the maximal route discovery time is $t_r = 7.120$ s.

For example, the following scenario is defined: node $n01$ pings node $n07$, the network topology and the start position of the additional node are depicted in figure 6.5. At some point in time, node $n06$ fails and the network splits into to partitions: $\{n01, n02, n03, n04, n05\}$ and $\{n07\}$. Note that the additional node belongs to the same partition as node $n01$ belongs to. The ground speed of the additional node is about 80 km/h (maximal speed of a multicopter used in demonstrator as an additional node) or about 22 m/s. The distance from the start position to the destination according to this scenario is about 477 m, so the movement time is about $t_m = 477/22 = 21$ s. A rough estimation of the recovery time gives $t_r + t_m = 28.120$ s. Emulation of this scenario produces the following time intervals: $t_r^{emu} = 7.491$ s, $t_m^{emu} = 19.739$ s and the recovery time 27.230 s. The recovery time is lower then the theoretically calculated (28.120 s) because communication was possible already before the additional node has reached its destination.

After that the functionality of the network partitioning recovery system was validated in the testbed, the next validation phase for the system is its testing in a real environment on real devices. The testing scenario is a part of the joint node placement demonstrator of the graduate school, and it will be presented in the following section in detail.

6.3 Implementation for Demonstrator

In order to demonstrate practical results of several PhD theses, a joint demonstrator was developed in the graduate school. The demonstrator combines the following research topics: localization of wireless nodes, autonomously flying Unmanned Aerial Vehicles (UAVs) and network partitioning recovery. The demonstrator is aimed to simulate the network recovery after a disaster.

The starting point of the demonstrator is a MANET with enabled multi-hop communication. The network is presented in the demonstrator by a chain of five netbooks, which communicate using the AODV routing protocol. The communication is shown by ping between first and fifth netbooks in the chain. According to the legend, an intermediate node in this chain was destroyed so that no more communication is possible. The network should be repaired using a UAV (a multicopter, MC) which is controlled from the disaster recovery headquarter via a server. The demonstrator is illustrated in figure 6.7.

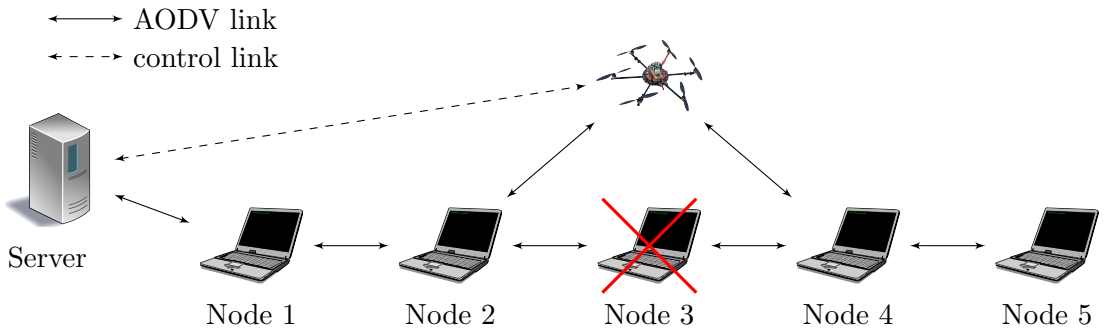


Figure 6.7: Demonstrator network architecture

The demonstration scenario can be described as a sequence of the following steps, which are shown in the message-sequence-chart in figure 6.8:

1. The network is working well.

2. One of nodes 2, 3 or 4 is destroyed (the netbook is switched off).

3. The network recognizes the communication break and notifies the headquarter server, which immediately initiates a localization mission for the MC.

4. The MC performs the localization mission by flying with a zig-zag pattern over the network area. During the flight the measurement data needed for the localization are sent periodically to the server, and the network nodes are being localized on-line.

5. After the localization mission was done and the nodes' positions were calculated, the server calculates the placement position using the Voronoi diagram-based approach.

6. Next, the server generates a mission request for the placement of the additional node (MC).

7. When the MC reaches the goal position, it switches on its network interface which is configured to be used with the AODV routing and participates the network.

8. After that a new route between nodes 1 and 5 is found, the communication is possible again and the recovery is done.

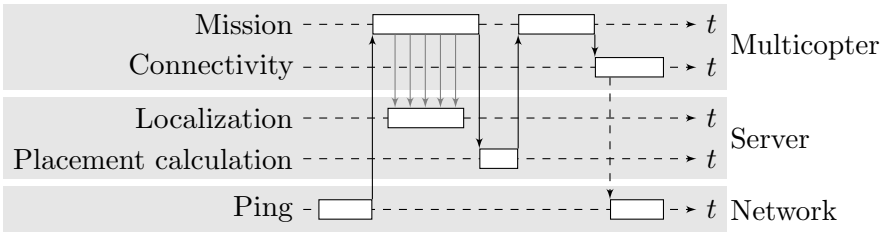


Figure 6.8: Scenario for the node placement demonstrator

Additionally, the demonstrator provides a GUI, in which the localization and the placement processes can be shown on-line on a map. GUI uses the GoogleEarth browser plugin as a front-end and a PHP script connected to the server database as a back-end.

In the following sections, the separate parts of the demo – the server, the network and the MC – will be presented in detail.

6.3.1 Network

As mentioned before, the test network consists of a chain of five netbooks. On the netbooks, the Linux operating system is running, and the netbooks are interconnected using WiFi in ad-hoc mode. The addresses are assigned statically, and multi-hop routing is supported by the AODV-Click implementation. According to the defined network topology, a node of the network can communicate directly only with its neighbors in the chain, otherwise multi-hop routing is used.

To assure the defined topology and to keep the area needed for the demonstrator smaller, the radio range of the nodes should be reduced, otherwise node 1 could communicate

with node 2 directly or the chain length should be more than $(5-1) \cdot 150 = 600$ meters. The first approach to reduce the communication range was to assign the minimum possible value for transmitter power for internal WiFi adapters using the *iwconfig* configuration utility. Experiments showed that the communication range was not changing though the specified values of the transmitter power were shown in the parameters of the network adapters.

The second approach was the usage of USB network adapters with external antennas. In order to reduce the communication radius, antennas were damped down. It allowed to reduce the communication radius up to 30 m, however it was unstable in time: sometimes node 1 was able to communicate with node 5 directly, and sometimes node 1 could not see its direct neighbor. Such a behavior breaks down the AODV routing mechanism and makes any communication impossible. The reason of this instability possibly lies in the driver of the USB adapters which does not allow to disable the power management mechanism, and it stays always active.

The last decision was to reduce the node ranges on the MAC layer by filtering of the packets according to their source MAC addresses. However, it makes the network topology hard coded and requires that the nodes in the chain should be always in the same order, but it allows to use the network within a small area and even in one room.

Additionally for the network nodes, a problem detection mechanism similar to the problem detector described in section 5.2.3 was implemented. The difference of the problem detector implemented for the demonstrator consists in that, rather the initialization of NNS, it generates an empty UDP packet with the specified destination IP-address and port number. This packet addresses the headquarter server and the trigger event, which starts the localization process for the nodes of the network to be repaired.

In order to automatize the network deployment, start and stop scripts were implemented. The scripts are responsible for the proper configuration of the network adapters in Linux, for configuration of the internal routing tables and for the start and termination of the AODV-Click scripts.

6.3.2 Multicopter

As a mobile freely movable node in the demonstrator, a multicopter (s. photo in figure 6.9) is used. The multicopter presents a mobile communication platform with the ability of autonomously navigation. Navigation is based on a GPS receiver supported by others sensors (tilt sensor, accelerometer, gyroscopes, ultrasonic sensor). The cooperation of these sensors allows a more accurate navigation than by using GPS only. The navigation system supports autonomous flights to a given goal position and flights around a specified trajectory including starting and landing.

The hardware platform of the multicopter is based on the Mikrokopter [SRWK12] chassis with six motors under the control of a Gumstix [Gum12] based board computer.

Figure 6.9: Multicopter

As communication platform, the multicopter is equipped with two wireless network (WiFi) interface devices. Two network interfaces make it possible to decouple the recovery and the usage communication planes. So, one interface is responsible for the control and measurement communications within the recovery network, and the other interface is used for the communications within the network being repaired. Additionally, the multicopter provides a radio link for remote control, which plays the role of a backup link. Via this link, a manual control of the multicopter can be taken over in an emergency situation.

From the software point of view, the multicopter is controlled by a real-time Linux operating system. All time critical applications and daemons are implemented using C/C++ and incapsulated in separate components which are interconnected via the SCL interface. Other software components, which are non real-time applications, are implemented with python but also use the SCL interface for the communication with other components. In order to provide the functionality of a node belonging to the repaired network, the AODV-Click script is deployed also on the multicopter and bound to the second network interface.

Within the demonstrator, the multicopter performs two types of missions. Firstly, it collects measurement data for localization: over the second network interface, the multicupter measures the received signal strength from the nodes of the network and sends the results together with the actual GPS coordinates and the time stamp to the server over the control link (the first network device). Secondly, the multicopter plays a role of the additional node for the network partitioning recovery system. On request of the server, it flies to a specified position in order to reconnect the network.

As soon as the mulitcopter achieves the destination position, it activates its second network interface and enables AODV on it, so that the multicopter can participate in the network and can take part in the routing. From the network's point of view, the multicopter is a normal node, the same as all other nodes.

6.3.3 Server

According to the demonstrator's scenario, the server belongs to the headquarter of the disaster recovery forces. In the demonstrator, the server is a notebook with dedicated tasks and software. The server is based on a Linux platform, and its architecture is depicted on figure 6.10.

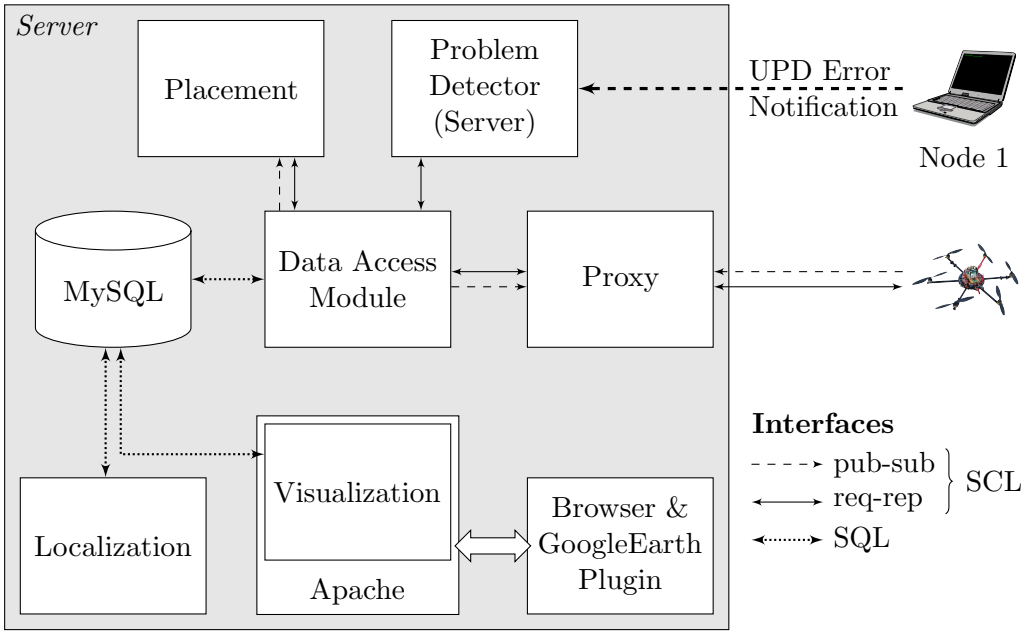


Figure 6.10: Demonstrator headquarter server architecture

All the server components are implemented as daemons and run in background. Like the multicopter, the server is equipped with two network interfaces: one for the control link with the multicopter and the other for the communication with the network to be repaired. For the second link, AODV-based communication is enabled. Via this link, the network can notify the server about a communication problem. For the notification, an empty UDP packet is sent out by node 1 to a specified address (the server address) and port number as described above.

The notification is accepted by the server-side part of the *Problem Detector*, which generates the localization mission request. Over the *Data Access Module* (*DAM*), the mission request is stored into the *MySQL* database and simultaneously sent to

the multicopter via the *Proxy*. The *Proxy* provides an interface between the SCL-based infrastructure of the server and the multicopter. There are two parallel links between the server and the multicopter: the request-reply and the publish-subscribe link. The first link is used for the mission requests, and via the second link the multicopter sends mission status updates and the measurement data (both do not need any acknowledgement from the server-side).

During the localization mission, the measurement data are being sent by the multicopter. Through the *Proxy* and *DAM*, the data is stored into the database. The *Localization* module implemented with PHP periodically reads the measurement data and updates the localized positions for the nodes of the network. In parallel, the measurement process can be visualized on-line using the *Visualization* PHP script running on the *Apache* [BFH+12] web-server. The localization data is shown in a browser on a map provided by the GoogleEarth [Goo12] browser plugin.

As soon as the localization mission is finished, the multicopter sends the mission status update with the status *done*. Through DAM, the *Placement* module is notified about the end of the localization mission and starts the placement calculation using the Voronoi diagram approach based on the data calculated by the *Localization* component. After that the placement position was calculated, the *Placement* component generates the placement mission request, which is forwarded to the multicopter, and the multicopter flies to the specified position in order to reconnect the network.

At the moment of writing of this work, the demonstrator is still in the development phase. Tests of individual components (nodes localization, calculation of target placement position) were completed successfully. The demonstrator as a whole is expected to be ready in summer 2013.

6.4 Summary

In this chapter, three phases of the simulation and evaluation of the network partitioning recovery system were presented. The first phase considered the simulation of the individual system components previously discussed in section 5. The goal of this phase is selecting and testing suitable algorithms and their parameters. The next phase consisted of the deployment of the system on a real Linux platform with an artificial environment – in the testbed. The purpose of this phase was the testing of the components' interaction and debugging the system as a whole. The final phase of the evaluation encompassed the testing of the system on a real Linux platform within a real environment. The system in configuration used in this phase is also a part of the joint network recovery demonstrator of the graduate school.

The following chapter will conclude this work and also discuss some problems and future work considering the network partitioning recovery system presented here.

7 Summary

The main focus of this work was set on the recovery of communication networks in disaster scenarios. The work was motivated primarily by the fact that a properly working communication network deployed in a disaster area can help to save more human lives because it supports coordination of the members of rescue teams and it enables emergency communications for standard user devices like smart phones and notebooks.

The most suitable network structure supporting communications in emergency situations is a mobile ad-hoc network (MANET). Such a network does not require any terrestrial infrastructure elements which are most vulnerable during disaster. A MANET is a self-organizing network without any central control entity, which would be a single point of failure. However, due to the nodes' mobility, a MANET can provide only a short- to middle-term solution, but exactly the first hours after disaster are the most important regarding life rescue. An overview of MANETs was given in chapter 2.

Since one of the typical problems of MANETs is network partitioning, which results from the mobility of nodes, a system for network partitioning recovery is needed to provide uninterrupted work of the network. Of course, the usage scenario of such a system within a disaster adds some extra requirements to this system, which were formulated in section 3.3. The systems known from literature, which were discussed in the same chapter in section 3.4, satisfy these requirements only partially.

The system presented in this work was designed with respect to the defined requirements and has the following advantages in comparison to the previously known system: the system is self-organizing and distributed (but it can utilize the advantages of a centralized approach, if the necessary infrastructure and information are already or still available in the network). The system exploits the idea of placement of additional nodes and does not require the network participants to change their positions. The system has reactive behavior and does not require additional communications if the network is connected. The complete system architecture as set of interconnected components was presented in chapter 4.

Each system component provides a particular function like partitioning detection or mission request. Obviously, there are many ways to implement each of these functions. The algorithms' comparison and selection are presented in chapter 5 together with the inter-component interface specification.

In the last chapter 6, the evaluation results of the system were discussed. The validation was split into three phases : simulation, emulation and hardware implementation in a demonstrator. The simulation phase allowed to test and select particular algorithms for separate system components. The second phase – emulation in a testbed – was focused on the cooperation and the interaction between components. Finally, the system (the variant with available global knowledge) was tested in a demonstrator.

7.1 Unique Benefits of the System

In this work, a network partitioning recovery system for application in disaster scenarios was developed. The system is able to work in both centralized and distributed modes demonstrating significant benefits in comparison to the systems described in [DPS08] and [CDB04], that utilize the same idea of placement of additional nodes but only with centralized control as discussed in section 3.4. The system does not require the rearrangement of the nodes participating in the network to be repaired, as it is done in systems [ASTU10, WL09, RK04, GLM+04, AT04], and because of that is suitable for the applications addressed by this work. The system also is more cost effective in comparison to the systems from [DPH05] and [MCG09] due to potentially smaller number of additional devices being used, and allows real-time data communication in contrast to the system utilizing the message ferrying idea [WL10].

The concept of the system proposed in this work is not restricted to any concrete wireless network technology used in the network to be repaired due to the separation into two logical communication planes: the repair plane and the usage plane as discussed in section 4.1.4. Thus, the same system concept can be used for the recovery of cellular network, if the additional nodes used by the system are equipped with mobile base stations.

7.2 Future Work

There are several ways to improve the concept presented here and to add some features which were out of the scope of this work.

The first idea is to embed some self-configuring methods to allow the system to switch between the two modes – centralized and distributed – autonomously depending on the active network structure. For this purpose, a mechanism recognizing the network structure should be developed and implemented. The main task of such a component is to detect whether global knowledge about the network topology is available for the network partitioning recovery system or not. Of course, the data format, in which the network topology is presented, should be known to the system.

Secondly, the subsystem responsible for placement position calculation could take into account additional geographical information about the network area, which allows to find better positions for additional nodes and, first of all, to detect possible obstacles which can hinder the placement.

Then, the placement possibilities for multiple additional nodes were not considered here and were referred to the mission planning system, which is not within the scope of this work.

Since the system is aimed to work with standard user devices like notebooks and smart phones, the implementation of the system for other software platforms, e.g. Windows, Android, Mac, should be considered, too.

And last but not least, some security mechanisms should be integrated into the system. Until now, no security issues have been considered, but they play an important role in real application scenarios. However, due to a large number of potential vulnerabilities for the system, the theme of security should be considered in a completely separate PhD topic.

Summarized, the network partitioning recovery system presented in this work fulfills all requirements defined in section 3.3. The main solution ideas behind this work were partially described in publications [Tar10] and [Tar12]. The achieved results are validated on a testbed, and the centralized variant of this system is a part of the joint network recovery demonstrator of the graduate school Mobicom.

Appendix

Bibliography

[AK00] F. Aurenhammer and R. Klein, "Voronoi diagrams," in *Handbook of Computational Geometry, Chapter V*, J. Sack and G. Urrutia, Eds. Elsevier Science Publishing, 2000, pp. 201–290, [SFB Report F003-092, TU Graz, Austria, 1996].

[AK08] P. Appavoo and K. Khedo, "SENCAST: A Scalable Protocol for Unicasting and Multicasting in a Large Ad hoc Emergency Network," *International Journal of Computer Science and Network Security*, vol. 8, no. 2, pp. 154–165, Feb. 2008. [Online]. Available: http://paper.ijcsns.org/07_book/200802/20080221.pdf

[ASM+11] O. Artemenko, G. Schorcht, S. Mann, D. Schulz, and M. Tarasov, "A Groupwise Universal Improvement Scheme for Location Estimation in Wireless Networks," in *Computer Communications and Networks (ICCCN), 2011 Proceedings of 20th International Conference on*, 31 2011-aug. 4 2011, pp. 1–5.

[AST10] O. Artemenko, G. Schorcht, and M. Tarasov, "A refinement scheme for location estimation process in indoor wireless sensor networks," in *GLOBECOM Workshops (GC Wkshps), 2010 IEEE*, dec. 2010, pp. 225–229.

[ASTU10] K. Akkaya, F. Senel, A. Thimmapuram, and S. Uludag, "Distributed Recovery from Network Partitioning in Movable Sensor/Actor Networks via Controlled Mobility," *Computers, IEEE Transactions on*, vol. 59, no. 2, pp. 258–271, feb. 2010.

[AT04] H. Asahi and K. Takahata, "Recovery protocol for dynamic network reconstruction on disaster information system," in *Advanced Information Networking and Applications, 2004. AINA 2004. 18th International Conference on*, vol. 2, Mar. 2004, pp. 87–90 Vol.2.

[AZM05] K. K. Al-Zahid and M. Matsumoto, "QoS aware routing for ad-hoc wireless network," in *Proceedings of the 4th international conference on Mobile and ubiquitous multimedia*, ser. MUM '05. New York, NY, USA: ACM, 2005, pp. 43–47. [Online]. Available: http://doi.acm.org/10.1145/1149488.1149496

[B. 05] B. Braem, "Implementation and evaluation of ad-hoc distance vector routing," Master's thesis, University of Antwerp, Jun. 2005. [Online]. Available: http://euterpe.cmi.ua.ac.be/~bbraem/thesis/thesis.pdf

[BC08] M. Bakhouya and N. Cottin, "Performance Evaluation of the Location-Based Protocol DREAM for Large Mobile Ad Hoc Networks," in *New Technologies, Mobility and Security, 2008. NTMS '08.*, 5-7 2008, pp. 1–6.

[BCSW98] S. Basagni, I. Chlamtac, V. R. Syrotiuk, and B. A. Woodward, "A distance routing effect algorithm for mobility (DREAM)," in *MobiCom '98: Proceedings of the 4th annual ACM/IEEE international conference on Mobile computing and networking*. New York, NY, USA: ACM Press, 1998, pp. 76–84. [Online]. Available: http://dx.doi.org/10.1145/288235.288254

[Bel58] R. Bellman, "On a Routing Problem," *Quarterly of Applied Mathematics*, vol. 16, pp. 87–90, 1958.

[BFH+12] B. Behlendorf, R. T. Fielding, R. Hartill *et al.* (2012) Apache HTTP Server Project. Project's homepage. The Apache Software Foundation. [Online]. Available: http://httpd.apache.org

[BLG05] L. Blazevic, J.-Y. Le Boudec, and S. Giordano, "A location-based routing method for mobile ad hoc networks," *Mobile Computing, IEEE Transactions on*, vol. 4, no. 2, pp. 97–110, march-april 2005.

[BMSU99] P. Bose, P. Morin, I. Stojmenović, and J. Urrutia, "Routing with guaranteed delivery in ad hoc wireless networks," in *DIALM '99: Proceedings of the 3rd international workshop on Discrete algorithms and methods for mobile computing and communications*. New York, NY, USA: ACM, 1999, pp. 48–55.

[BNSM09] M. Bakhouya and A. Nait-Sidi-Moh, *Localisation et Routage Géographique dans les Réseaux Mobiles Ad Hocà Large Echelle*. PPUR-UTBM, 2009, ch. Position-based Routing.

[BR08] R. J. Barton and D. Rao, "Performance capabilities of long-range UWB-IR TDOA localization systems," *EURASIP J. Adv. Signal Process*, vol. 2008, pp. 81:1–81:17, Jan. 2008. [Online]. Available: http://dx.doi.org/10.1155/2008/236791

[Bra89] R. Braden, "Requirements for Internet Hosts – Communication Layers," RFC-1122, IETF Network Working Group, Oct. 1989. [Online]. Available: http://tools.ietf.org/html/rfc1122

[Bra09] B. Braem. (2009) An implementation of AODV in Click. PATS research group at University of Antwerp. [Online]. Available: http://www.pats.ua.ac.be/software/aodv

[BST03] C. Busch, S. Surapaneni, and S. Tirthapura, "Analysis of link reversal routing algorithms for mobile ad hoc networks," in *Proceedings of the fifteenth annual ACM symposium on Parallel algorithms and architectures*, ser. SPAA '03. New York, NY, USA: ACM, 2003, pp. 210–219. [Online]. Available: http://doi.acm.org/10.1145/777412.777446

[BT05] J. Bachrach and C. Taylor, "Localization in Sensor Networks," *Artificial Intelligence*, pp. 327–349, 2005. [Online]. Available: http://portal.acm.org/citation.cfm?id=1013825.1013841

[BZGV08] R. J. Barton, R. L. Zheng, S. Gezici, and V. V. Veeravalli, "Signal Processing for Location Estimation and Tracking in Wireless Environments," *EURASIP J. Adv. Sig. Proc.*, vol. 2008, 2008.

[C+11] B. Carpenter *et al.*, "IP Option Numbers," IANA, 2011. [Online]. Available: http://www.iana.org/assignments/ip-parameters

[C+12] G. Combs *et al.* (2012) Wireshark. Wireshark Foundation. [Online]. Available: http://www.wireshark.org

[Cam03] S. Camazine, "Self-organizing Systems," *Encyclopedia of cognitive science*, pp. 1059–1062, 2003. [Online]. Available: http://web.mac.com/camazine/Camazine/Self-organization_files/Self-organization.pdf

[CCT03] C.-Y. Chang, C.-T. Chang, and S.-C. Tu, "Obstacle-free geocasting protocol for ad hoc wireless networks," in *Wireless Communications and Networking, 2003. WCNC 2003. 2003 IEEE*, vol. 2, 20-20 2003, pp. 1137–1142 vol.2.

[CDB04] K. Chandrashekar, M. Dekhordi, and J. Baras, "Providing full connectivity in large ad-hoc networks by dynamic placement of aerial platforms," in *Military Communications Conference, 2004. MILCOM 2004. 2004 IEEE*, vol. 3, oct.-3 nov. 2004, pp. 1429–1436 Vol. 3.

[Chi11] *BeiDou Navigation Satellite System Signal in Space Interface Control Document*, China Satellite Navigation Office Std., Dec. 2011.

[Chr11] J. Chroboczek, "The Babel Routing Protocol," RFC-6126, PPS, University of Paris, Apr. 2011. [Online]. Available: http://www.rfc-editor.org/rfc/rfc6126.txt

[CJ03] T. Clausen and P. Jacquet, "Optimized Link State Routing Protocol (OLSR)," RFC-3626, IETF Network Working Group, Oct. 2003. [Online]. Available: http://www.ietf.org/rfc/rfc3626.txt

[CL03] T. Camp and Y. Liu, "An adaptive mesh-based protocol for geocast routing," *J. Parallel Distrib. Comput.*, vol. 63, no. 2, pp. 196–213, 2003.

[CLR09] T. H. Cormen, C. E. Leiserson, and R. L. Rivest, *Introduction to Algorithms*, 3rd ed. MIT Press and McGraw-Hill, 2009, ch. 24.1: The Bellman–Ford algorithm, pp. 651–655.

[CSAB08] D. Conan, P. Sens, L. Arantes, and M. Bouillaguet, "Failure, Disconnection and Partition Detection in Mobile Environment," in *NCA '08: Proceedings of the 2008 Seventh IEEE International Symposium on Network Computing and Applications*. Washington, DC, USA: IEEE Computer Society, 2008, pp. 119–127.

[Deb09] M. Debes, "Konzeption und Realisierung eines kontextsensitiven Routingverfahrens," Ph.D. dissertation, Ilmenau University of Technology, Ilmenau, Germany, 2009. [Online]. Available: http://www.db-thueringen.de/servlets/DerivateServlet/Derivate-15887

[DH98] S. Deering and R. Hinden, "RFC 2460 Internet Protocol, Version 6 (IPv6) Specification," Internet Engineering Task Force, December 1998. [Online]. Available: http://tools.ietf.org/html/rfc2460

[Dij59] E. W. Dijkstra, "A Note on Two Problems in Connexion with Graphs," *Numerische Mathematik*, vol. 1, no. 1, pp. 269–271, 1959. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.165.7577

[DPH05] S. Das, H. Pucha, and Y. Hu, "Microrouting: a scalable and robust communication paradigm for sparse ad-hoc networks," in *Parallel and Distributed Processing Symposium, 2005. Proceedings. 19th IEEE International*, april 2005, p. 8 pp.

[DPS08] G. Dini, M. Pelagatti, and I. M. Savino, "An algorithm for reconnecting wireless sensor network partitions," in *Proceedings of the 5th European conference on Wireless sensor networks*, ser. EWSN'08. Berlin, Heidelberg: Springer-Verlag, 2008, pp. 253–267. [Online]. Available: http://dl.acm.org/citation.cfm?id=1786014.1786036

[ECC03] J. Esteves, A. Carvalho, and C. Couto, "Generalized geometric triangulation algorithm for mobile robot absolute self-localization," in *Industrial Electronics, 2003. ISIE '03. 2003 IEEE International Symposium on*, vol. 1, june 2003, pp. 346–351 vol. 1.

[EUR98] *WGS 84 Implementation Manual*, EUROCONTROL European Organization for the Safety of Air Navigation Brussels, Belgium and IfEN Institute of Geodesy and Navigation (IfEN) University FAF Munich, Germany Std., Feb. 1998. [Online]. Available: http://www.dqts.net/files/wgsman24.pdf

[FDKC06] T. Fuhrmann, P. Di, K. Kutzner, and C. Cramer, "Pushing Chord into the Underlay: Scalable Routing for Hybrid MANETs," System Architecture Group, Universität Karlsruhe (TH), Tech. Rep. 2006-12, Jun. 2006. [Online]. Available: http://www.so.in.tum.de/publications/pdfs/fuhrmann06pushing.pdf

[FF56] L. R. Ford and D. R. Fulkerson, "Maximal Flow through a Network." *Canadian Journal of Mathematics*, vol. 8, pp. 399–404, 1956. [Online]. Available: http://www.rand.org/pubs/papers/P605

[FHR+11] B. Fenner, G. Harris, M. Richardson *et al.* (2011) Tcpdump packet analyser. [Online]. Available: http://www.tcpdump.org

[FLS06] K.-W. Fan, S. Liu, and P. Sinha, *Handbook of Algorithms for Wireless Networking and Mobile Computing*, ser. Chapman & Hall/CRC Computer and Information Science. Chapman & Hall/CRC and Taylor & Francis Group, 2006, ch. 9: Ad Hoc Routing Protocols, pp. 183–215.

[FO05] S. Fortune and S. O'Sullivan, "Sweep line algorithm (Fortune's algorithm) for the calculation of Voronoi-diagram," 2005. [Online]. Available: http://www.skynet.ie/~sos/mapviewer/voronoi.php

[For87] S. Fortune, "A Sweepline Algorithm for Voronoi Diagrams," *Algorithmica*, vol. 2, pp. 153–174, 1987.

[Gal12] "Galileo," 2012. [Online]. Available: http://www.satellite-navigation.eu

[GCWI08] I. Güvenç, C.-C. Chong, F. Watanabe, and H. Inamura, "NLOS identification and weighted least-squares localization for UWB systems using multipath channel statistics," *EURASIP J. Adv. Signal Process*, vol. 2008, Jan. 2008. [Online]. Available: http://dx.doi.org/10.1155/2008/271984

[GLM+04] D. K. Goldenberg, J. Lin, A. S. Morse, B. E. Rosen, and Y. R. Yang, "Towards mobility as a network control primitive," in *Proceedings of the 5th ACM international symposium on Mobile ad hoc networking and computing*, ser. MobiHoc '04. New York, NY, USA: ACM, 2004, pp. 163–174. [Online]. Available: http://doi.acm.org/10.1145/989459.989481

[GLO12] "GLONASS – Global Navigation Satellite System," 2012. [Online]. Available: http://www.glonass-ianc.rsa.ru

[Goo12] (2012) Google Earth Plugin. Project's homepage. Google. [Online]. Available: http://www.google.de/earth/explore/products/plugin.html

[GPS12] "GPS – Global Positioning System," 2012. [Online]. Available: http://www.gps.gov

[Gri08] J. Grimes, *Global Positioning System Standard Positioning Service Performance Standard*, Department of Defence USA, NAVSTAR-GPS Std., Sep. 2008. [Online]. Available: http://www.gps.gov/technical/ps/2008-SPS-performance-standard.pdf

[Gro99] O. M. Group, "The Common Object Request Broker: Architecture and Specification (CORBA 2.3.1 specification)," Object Management Group, Tech. Rep., Oct. 1999.

[GS69] K. R. Gabriel and R. R. Sokal, "A New Statistical Approach to Geographic Variation Analysis," *Systematic Biology*, vol. 18, no. 3, pp. 259–278, 1969. [Online]. Available: http://sysbio.oxfordjournals.org/content/18/3/259.abstract

[GS78] P. J. Green and R. Sibson, "Computing Dirichlet Tessellations in the Plane," *Computer Journal*, vol. 21, no. 2, pp. 168–173, 1978.

[Gum12] (2012) Gumstix. web site. Gumstix inc. [Online]. Available: http://gumstix.com

[HCS03] M. Hauspie, J. Carle, and D. Simplot, "Partition Detection in Mobile Ad-Hoc Networks Using Multiple Disjoint Paths Set," in *in Proceedings of 2nd IFIP Mediterranean Ad Hoc Networking Workshop (MED-HOC-NET)*, 2003, pp. 25–27.

[Hen10a] T. Henderson, "NS-3 Tutorial: Part 1," 9th GENI Engineering Conference (GEC9), p. 27, nov 2010. [Online]. Available: http://www.nsnam.org/tutorials/geni-tutorial-part1.pdf

[Hen10b] V. M. Henze, "MANET-Routing: Klassifizierung und Vergleich von Routing-Protokollen," Advanced seminar work, Ilmenau University of Technology, Communication Networks Research Lab, Jul. 2010, in German. [Online]. Available: http://yukon.e-technik.tu-ilmenau.de/~webkn/Abschlussarbeiten/Hauptseminararbeiten/hs_henze.pdf

[HKCW+97] C.-C. C. Hsiao-Kuang, C.-c. Chiang, H.-k. Wu, W. Liu, and M. Gerla, "Routing In Clustered Multihop, Mobile Wireless Networks With Fading Channel," in *IEEE Singapore International Conference on Networks*, 1997. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.50.8359

[HLP07] S.-P. Hong, C.-S. Lee, and J.-A. Park, "Design of Variable Geocasting Based on Ad-Hoc Networks," in *Wireless Communications, Networking and Mobile Computing, 2007. WiCom 2007. International Conference on*, 21-25 2007, pp. 1735–1738.

[HM06] L. Hughes and A. Maghsoudlou, "An efficient coverage-based flooding scheme for geocasting in mobile ad hoc networks," in *Advanced Information Networking and Applications, 2006. AINA 2006. 20th International Conference on*, vol. 1, 18-20 2006, p. 6 pp.

[HOM10] A. Huang, E. Olson, and D. Moore, "LCM: Lightweight Communications and Marshalling," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, October 2010.

[Hor08] R. Hornig, "INETMANET Framework for OMNEST/OMNeT++ 4.x (based on INET Framework)," 2008. [Online]. Available: https://github.com/inetmanet/inetmanet/wiki

[HPS02] Z. J. Haas, M. R. Pearlman, and P. Samar, *The Zone Routing Protocol (ZRP) for Ad Hoc Networks*, IETF Mobile Ad hoc Networks Working Group Internet-Draft, Jul. 2002. [Online]. Available: http://tools.ietf.org/html/draft-ietf-manet-zone-zrp

[HRFR06] T. R. Henderson, S. Roy, S. Floyd, and G. F. Riley, "NS-3 project goals," in *WNS2 '06: Proceeding from the 2006 workshop on ns-2: the IP network simulator*. New York, NY, USA: ACM, 2006, pp. 13+. [Online]. Available: http://dx.doi.org/10.1145/1190455.1190468

[HSZ+12] G. Han, W. Shen, C. Zhu, L. Shu, and J. J. Rodrigues, "RNST: Precise Localization Based on Trilateration for Indoor Sensor Networks," in *Advancements in Distributed Computing and Internet Technologies: Trends and Issues*. IGI Global, 2012, pp. 230–257.

[HV02] G. Holland and N. H. Vaidya, "Analysis of TCP Performance over Mobile Ad hoc Networks," *ACM/Kluwer Journal of Wireless Networks*, vol. 8, pp. 275–288, 2002.

[IPD11] V. Iyer, A. Pruteanu, and S. Dulman, "NetDetect: Neighborhood Discovery in Wireless Networks Using Adaptive Beacons," in *Proceedings of the 2011 IEEE Fifth International Conference on Self-Adaptive and Self-Organizing Systems*, ser. SASO '11. Washington, DC, USA: IEEE Computer Society, 2011, pp. 31–40. [Online]. Available: http://dx.doi.org/10.1109/SASO.2011.14

[JC02] X. Jiang and T. Camp, "A Review of Geocasting Protocols for a Mobile Ad Hoc Network," Proc. Grace Hopper Celebration (GHC), 2002.

[JHM07] D. Johnson, Y. Hu, and D. Maltz, "The Dynamic Source Routing Protocol (DSR) for Mobile Ad Hoc Networks for IPv4," RFC-4728, IETF Network Working Group, Feb. 2007. [Online]. Available: http://www.ietf.org/rfc/rfc4728.txt

[JK07] P. Jeon and G. Kesidis, "GeoPPRA: An Energy-Efficient Geocasting Protocol in Mobile Ad Hoc Networks," in *Networking, 2007. ICN '07. Sixth International Conference on*, 22-28 2007, pp. 10–10.

[JM96] D. B. Johnson and D. A. Maltz, "Dynamic source routing in ad hoc wireless networks," in *Mobile Computing*. Kluwer Academic Publishers, 1996, pp. 153–181. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.129.426

[JPDG04] A. Jayasuriya, S. Perreau, A. Dadej, and S. Gordon, "Hidden vs. exposed terminal problem in ad hoc networks," in *Proceedings of the Australian Telecommunication Networks and Applications Conference, Sydney, Australia*, 2004.

[Kar00] B. N. Karp, "Geographic routing for wireless networks," Ph.D. dissertation, Cambridge, MA, USA, 2000, adviser-Kung, H. T.

[KGKS05] Y.-J. Kim, R. Govindan, B. Karp, and S. Shenker, "Geographic routing made practical," in *NSDI'05: Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation*. Berkeley, CA, USA: USENIX Association, 2005, pp. 217–230.

[KK00] B. Karp and H. T. Kung, "GPSR: greedy perimeter stateless routing for wireless networks," in *MobiCom '00: Proceedings of the 6th annual international conference on Mobile computing and networking*. New York, NY, USA: ACM, 2000, pp. 243–254.

[KMC+00] E. Kohler, R. Morris, B. Chen, J. Jannotti, and M. F. Kaashoek, "The Click modular router," *ACM Transactions on Computer Systems*, vol. 18, no. 3, pp. 263–297, August 2000.

[KU10] R. Kiruthika and R. Umarani, "An Exploration of Count-to-Infinity Problem in Networks," *International Journal of Engineering Science and Technology*, vol. 2, no. 12, pp. 7155–7159, 2010. [Online]. Available: http://www.ijest.info/docs/IJEST10-02-12-065.pdf

[KV98] Y.-B. Ko and N. H. Vaidya, "Location-Aided Routing (LAR) in Mobile Ad Hoc Networks," in *Mobile Computing and Networking, MOBICOM'98, October 25-30, 1998, Dallas, Texas, USA*. ACM / IEEE, October 1998, pp. 66–75. [Online]. Available: http://www.crhc.uiuc.edu/~nhv/old.papers/mobile-computing/mobicom98.pdf

[KV99] Y.-B. Ko and N. H. Vaidya, "Geocasting in mobile ad hoc networks: location-based multicast algorithms," in *Mobile Computing Systems and Applications, 1999. Proceedings. WMCSA '99. Second IEEE Workshop on*, 1999, pp. 101–110.

[KV00] Y.-B. Ko and N. H. Vaidya, "GeoTORA: a protocol for geocasting in mobile ad hoc networks," in *Network Protocols, 2000. 2000 International Conference on*, 2000, pp. 240–250.

[L+07] G. Lukas *et al.* (2007) AWDS: Ad-hoc Wireless Distribution Service. Otto-von-Guericke University Magdeburg. [Online]. Available: https://ivs-pm.ovgu.de/projects/awds

[L+11] D. Lezcano *et al.* (2006–2011) LXC Linux Containers. SourceForge Project. [Online]. Available: http://lxc.sourceforge.net

[LAB+11] A. Legout, E. Amir, H. Balakrishnan *et al.* (2011) The Network Simulator (NS-2). [Online]. Available: http://nsnam.isi.edu/nsnam/index.php/Main_Page

[LAOF05] L. Latiff, A. Ali, C.-C. Ooi, and N. Fisal, "Location-based geocasting and forwarding (LGF) routing protocol in mobile ad hoc network," in *Telecommunications, 2005. advanced industrial conference on telecommunications/service assurance with partial and intermittent resources conference/e-learning on telecommunications workshop. aict/sapir/elete 2005. proceedings*, 17-20 2005, pp. 536–541.

[LC05] J. J.-N. Liu and I. Chlamtac, *Mobile Ad Hoc Networking with a View of 4G Wireless: Imperatives and Challenges.* John Wiley & Sons, Inc., 2005, pp. 1–45.

[LCLM11] Y.-C. Liang, K.-C. Chen, G. Li, and P. Mahonen, "Cognitive Radio Networking and Communications: An Overview," *Vehicular Technology, IEEE Transactions on*, vol. 60, no. 7, pp. 3386–3407, sept. 2011.

[LJD+00] J. Li, J. Jannotti, D. S. J. De Couto, D. R. Karger, and R. Morris, "A Scalable Location Service for Geographic Ad Hoc Routing," in *Proceedings of the 6th ACM International Conference on Mobile Computing and Networking (MobiCom '00)*, Boston, Massachusetts, August 2000, pp. 120–130.

[LK06] S.-H. Lee and Y.-B. Ko, "Geometry-driven Scheme for Geocast Routing in Mobile Ad Hoc Networks," in *Vehicular Technology Conference, 2006. VTC 2006-Spring. IEEE 63rd*, vol. 2, 7-10 2006, pp. 638–642.

[LL08] H.-C. Liao and C.-J. Lin, "A Position-Based Connectionless Routing Algorithm for MANET and WiMAX under High Mobility and Various Node Densities," *Information Technology*, vol. 7, no. 3, pp. 458–465, 2008.

[LTS00] W.-H. Liao, Y.-C. Tseng, and J.-p. Sheu, "GeoGRID: A geocasting protocol for mobile ad hoc networks based on grid," *Journal of Internet Technology*, 2000.

[LTS01] W.-H. Liao, Y.-C. Tseng, and J.-P. Sheu, "GRID: A Fully Location-Aware Routing Protocol for Mobile Ad Hoc Networks," 2001.

[LW+12] M. Lacage, M. Weigle *et al.* (2012) NS-3 Simulator. [Online]. Available: http://www.nsnam.org

[LZH+06] M. J. Lee, J. Zheng, X. Hu, H.-h. Juan, C. Zhu, Y. Liu, J. S. Yoon, and T. Saadawi, "A new taxonomy of routing algorithms for wireless mobile ad hoc networks: the component approach," *Communications Magazine, IEEE*, vol. 44, no. 11, pp. 116–123, november 2006.

[MCG09] M. McClure, D. R. Corbett, and D. W. Gage, "The DARPA LANdroids program," in *Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series*, ser. Society of Photo-Optical Instrumentation Engineers (SPIE) Conference Series, vol. 7332, May 2009.

[MDP02] M. Maleki, K. Dantu, and M. Pedram, "Power-aware source routing protocol for mobile ad hoc networks," in *Low Power Electronics and Design, 2002. ISLPED '02. Proceedings of the 2002 International Symposium on*, 2002, pp. 72–75.

[MH96] S. A. M. Makki and G. Havas, "Distributed algorithms for depth-first search," *Information Processing Letters*, vol. 60, no. 1, pp. 7–12, 1996. [Online]. Available: http://www.sciencedirect.com/science/article/B6V0F-3VTK3D3-2/2/bf9e168e4939e28a411386d4b7cb8a78

[MK04] Z. Mir and S. Khan, "A Zone-based location service for geocasting in mobile ad hoc networks," in *Communications, 2004 and the 5th International Symposium on Multi-Dimensional Mobile Communications Proceedings. The 2004 Joint Conference of the 10th Asia-Pacific Conference on*, vol. 2, 29 2004, pp. 955–960 vol.2.

[MKHS09] M. Mubarik, S. Khan, S. Hassan, and N. Sarfraz, "Implementation of Geocast Enhanced AODV-UU in Linux Testbed," in *Computer and Information Science, 2009. ICIS 2009. Eighth IEEE/ACIS International Conference on*, 1-3 2009, pp. 803–807.

[MMS06] R. Mellier, J. Myoupo, and I. Sow, "GPS-Free Geocasting Algorithms in Mobile Ad hoc Networks," in *Wireless and Mobile Communications, 2006. ICWMC '06. International Conference on*, 29-31 2006, pp. 38–38.

[MR+12] P. McHardy, P. Russel *et al.*, "The netfilter.org project," 1999–2012. [Online]. Available: http://www.netfilter.org

[MRM09] F. Maymi and M. Rodriguez-Martinez, "Obstacle Avoidance for Utility-Based Geocasting," in *Information Technology: New Generations, 2009. ITNG '09. Sixth International Conference on*, 27-29 2009, pp. 338–343.

[Nel81] B. J. Nelson, "Remote procedure call," Ph.D. dissertation, Pittsburgh, PA, USA, 1981, aAI8204168.

[OF04] C.-C. Ooi and N. Fisal, "Implementation of geocast-enhanced AODV-bis routing protocol in MANET," in *TENCON 2004. 2004 IEEE Region 10 Conference*, vol. B, 21-24 2004, pp. 660–663 Vol. 2.

[OM84] M. I. T. Ohya and K. Murota, "Improvements of the incremental method for the Voronoi diagram with computational comparison of various algorithms," *Journal of the Operations Research Society of Japan*, no. 27, 1984.

[PAKG06] A. Pandey, M. Ahmed, N. Kumar, and P. Gupta, "A Hybrid Routing Scheme for Mobile Ad Hoc Networks with Mobile Backbones," in *High Performance Computing - HiPC 2006*, ser. Lecture Notes in Computer Science, Y. Robert, M. Parashar, R. Badrinath, and V. Prasanna, Eds. Springer Berlin / Heidelberg, 2006, vol. 4297, pp. 411–423, 10.1007/11945918_41. [Online]. Available: http://dx.doi.org/10.1007/11945918_41

[PC01] V. Park and S. Corson, "Temporally-Ordered Routing Algorithm (TORA). Version 1 Functional Specification," Internet draft, IETF MANET Working Group, Jul. 2001.

[PC12] C. Perkins and I. Chakeres, *Dynamic MANET On-demand (DYMO) Routing*, IETF Mobile Ad hoc Networks Working Group Internet-Draft, Mar. 2012. [Online]. Available: http://tools.ietf.org/html/draft-ietf-manet-dymo

[PGC00] G. Pei, M. Gerla, and T.-W. Chen, "Fisheye State Routing in Mobile Ad Hoc Networks," in *ICDCS Workshop on Wireless Networks and Mobile Computing*, 2000, pp. 71–78. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.43.6730

[PRC03] C. Perkins, E. Royer, and I. D. Chakeres. (2003, Oct.) Ad hoc On Demand Distance Vector (AODV) routing. IETF MANET Working Group Internet Draft. [Online]. Available: http://www.ietf.org/internet-drafts/draft-ietf-manet-aodvbis-00.txt

[PRD03] C. Perkins, E. Royer, and S. Das. (2003, Jul.) Ad hoc On Demand Distance Vector (AODV) routing. IETF Network Working Group Request for Comments: RFC 3561. [Online]. Available: http://www.ietf.org/rfc/rfc3561.txtinternet-drafts/draft-ietf-manet-aodv-13.txt

[pro12] "Google Protocol Buffers," Website, April 2012, available online: http://code.google.com/p/protobuf.

[QCG+09] M. Quigley, K. Conley, B. P. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Y. Ng, "ROS: an open-source Robot Operating System," in *ICRA Workshop on Open Source Software*, 2009.

[QLL] A. Quartulli, L. Lüssing, and M. Lindner. B.A.T.M.A.N. (Better Approach To Mobile Ad hoc Network). Open-mesh Project. [Online]. Available: http://www.open-mesh.org

[R+12] E. S. Raymond *et al.*, "gpsd – a GPS service daemon," 2005–2012. [Online]. Available: http://www.catb.org/gpsd

[RC08] M. Rajan and M. G. Chandra, "A Study on Network Partition Detection Relevant to Ad-hoc Networks: Connectivity Index Approach," *Tata Consultancy Services & Research Scholar Dept. of Computer Science, Dravidian University, Kuppam, India*, 2008.

[RFC81] "RFC 791 Internet Protocol - DARPA Inernet Programm, Protocol Specification," Internet Engineering Task Force, September 1981. [Online]. Available: http://tools.ietf.org/html/rfc791

[RK04] R. Rao and G. Kesidis, "Purposeful mobility for relaying and surveillance in mobile ad hoc sensor networks," *Mobile Computing, IEEE Transactions on*, vol. 3, no. 3, pp. 225–231, july-aug. 2004.

[RN09] S. Russell and P. Norvig, *Artificial Intelligence: A Modern Approach*, 3rd ed. Prentice Hall, Dec. 2009. [Online]. Available: http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20&path=ASIN/0136042597

[RWS08] H. Ritter, R. Winter, and J. Schiller, "A Partition Detection System for Mobile Ad-Hoc Networks," *Institute of Computer Science, Freie Universität Berlin, Germany*, p. 9, 2008.

[Sch12] S. Scheibe, "Erkennung und Behebung von Netzpartitionierungen in MANETs and WSNs," Ilmenau University of Technology, Communication Networks Research Lab, Tech. Rep., 2012, in German. [Online]. Available: http://yukon.e-technik.tu-ilmenau.de/~webkn/Abschlussarbeiten/Hauptseminararbeiten/hs_scheibe.pdf

[SH75] M. I. Shamos and D. Hoey, "Closest-point problems," in *Foundations of Computer Science, 1975., 16th Annual Symposium on*, Oct. 1975, pp. 151–162.

[SH04] K. Seada and A. Helmy, "Efficient geocasting with perfect delivery in wireless networks," in *Wireless Communications and Networking Conference, 2004. WCNC. 2004 IEEE*, vol. 4, 21-25 2004, pp. 2551–2556 Vol.4.

[SH06] K. Seada and A. Helmy, Eds., *Efficient and robust geocasting protocols for sensor networks*, vol. 29, no. 2, 2006, dependable Wireless Sensor Networks. [Online]. Available: http://www.sciencedirect.com/science/article/B6TYP-4GKWC1Y-1/2/ab113f4d71f79ed245e9c159f3b216fe

[SI92] K. Sugihara and M. Iri, "Construction of the Voronoi diagram for 'one million' generators in single-precision arithmetic," *Proceedings of the IEEE*, vol. 80, no. 9, pp. 1471–1484, Sep. 1992.

[Sim08] M. Simon, "Map-based Prediction of Network Partitioning in Wireless Sensor Networks," Master's thesis, Technical University of Darmstadt, 2008.

[SL99] I. Stojmenović and X. Lin, "GEDIR: Loop-Free Location Based Routing in Wireless Networks," in *IASTED International Conference on Parallel and Distributed Computing and Systems*, Boston, MA, USA, Nov. 1999, pp. 1025–1028.

[SL09] W.-C. Song and H. Lutfiyya, "Delivery-Guaranteed Geocast in MANETs by Using ZHLS," in *Computer Sciences and Convergence Information Technology, 2009. ICCIT '09. Fourth International Conference on*, 24-26 2009, pp. 86–90.

[SLG01] W. Su, S.-J. Lee, and M. Gerla, "Mobility prediction and routing in ad hoc wireless networks," *Int. J. Netw. Manag.*, vol. 11, no. 1, pp. 3–30, Jan. 2001. [Online]. Available: http://dx.doi.org/10.1002/nem.386

[SLS06] M. Sheng, J. Li, and Y. Shi, "Critical Nodes Detection in Mobile Ad Hoc Network," *Advanced Information Networking and Applications, International Conference on*, vol. 2, pp. 336–340, 2006.

[SPTK12] T. Simon, A. Puschmann, M. Tarasov, and S. N. Khan, "A Lightweight Message-based Inter-Component Communication Infrastructure," 2012, to be submitted.

[SRL03] I. Stojmenović, A. Ruhil, and D. Lobiyal, "Voronoi diagram and convex hull based geocasting and routing in wireless networks," in *Computers and Communication, 2003. (ISCC 2003). Proceedings. Eighth IEEE International Symposium on*, 30 2003, pp. 51–56 vol.1.

[SRWK12] T. Simon, J. Römisch, K. Wenzel, and A. Krause, "ARCADE: Airborne Robots for Communication and Autonomy in Disaster Environments," 2012. [Online]. Available: http://arcade-uav.github.com

[SSB98] R. Sivakumar, P. Sinha, and V. Bharghavan, "Core Extraction Distributed Ad hoc Routing (CEDAR) Specification," Internet draft, University of Illinois, Oct. 1998. [Online]. Available: http://tools.ietf.org/html/draft-ietf-manet-cedar-spec

[SST05] N. Shrivastava, S. Suri, and C. Toth, "Detecting cuts in sensor networks," in *Information Processing in Sensor Networks, 2005. IPSN 2005. Fourth International Symposium on*, Apr. 2005, pp. 210–217.

[TAGT08] O. Türkyilmaz, F. Alagöz, G. Gür, and T. Tugcu, "Environment-aware location estimation in cellular networks," *EURASIP J. Adv. Signal Process*, vol. 2008, pp. 139:1–139:9, Jan. 2008. [Online]. Available: http://dx.doi.org/10.1155/2008/276456

[Tar10] M. Tarasov, "Erkennung und Behebung von Netzpartitionierungen," in *11. Ilmenauer TK-Manager Workshop. Tagungsband.* Ilmenau University of Technology, Communication Networks Research Lab, 2010, pp. 77–84.

[Tar12] M. Tarasov, "Erkennung und Behebung von Netzpartitionierungen," in *12. Ilmenauer TK-Manager Workshop. Tagungsband.* Ilmenau University of Technology, Communication Networks Research Lab, 2012, pp. 121–128.

[TI10] S. Tanaka and D. C. Istiyanto, "Tsunami Hazard Mapping in Developing Countries: An effective way of raising awareness for tsunami disaster risk reduction," International Centre for Water Hazard and Risk Management under the auspices of UNESCO (ICHARM), Tech. Rep., Nov. 2010. [Online]. Available: http://www.icharm.pwri.go.jp/publication/pdf/2010/4184_tsunami_hazard_mapping.pdf

[Tou80] G. T. Toussaint, "The Relative Neighbourhood Graph of a Finite Planar Set," *Pattern Recognition*, vol. 12, pp. 261–268, 1980.

[TPQ08] H. Tang, Y. Park, and T. Qiu, "A TOA-AOA-based NLOS error mitigation method for location estimation," *EURASIP J. Adv. Signal Process*, vol. 2008, pp. 86:1–86:14, Jan. 2008. [Online]. Available: http://dx.doi.org/10.1155/2008/682528

[TSA11] M. Tarasov, J. Seitz, and O. Artemenko, "A network partitioning recovery process in Mobile Ad-Hoc Networks," in *Wireless and Mobile Computing, Networking and Communications (WiMob), 2011 IEEE 7th International Conference on*, oct. 2011, pp. 32–36.

[V+12] A. Varga *et al.* (2012) OMNeT++. Technical University of Budapest, Department of Telecommunications (BME-HIT). [Online]. Available: http://www.omnetpp.org

[VGLA04] R. Vaishampayan and J. Garcia-Luna-Aceves, "Efficient and robust multicast routing in mobile ad hoc networks," in *Mobile Ad-hoc and Sensor Systems, 2004 IEEE International Conference on*, oct. 2004, pp. 304–313.

[VH08] A. Varga and R. Hornig, "An overview of the OMNeT++ simulation environment," in *Simutools '08: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops*. ICST, Brussels, Belgium, Belgium: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering), 2008, pp. 1–10.

[VSR03] A. Valera, W. Seah, and S. Rao, "Cooperative packet caching and shortest multipath routing in mobile ad hoc networks," in *INFOCOM 2003. Twenty-Second Annual Joint Conference of the IEEE Computer and Communications. IEEE Societies*, vol. 1, march-3 april 2003, pp. 260–269 vol.1.

[WH07] M. Watanabe and H. Higaki, "No-Beacon GEDIR: Location-Based Ad-Hoc Routing with Less Communication Overhead," in *Information Technology, 2007. ITNG '07. Fourth International Conference on*, 2-4 2007, pp. 48–55.

[WL09] T. Wang and C. P. Low, "A Fully Distributed Node Allocation Scheme for Partition Protection in MANET," in *Wireless Communications and Networking Conference, 2009. WCNC 2009. IEEE*, april 2009, pp. 1–6.

[WL10] T. Wang and C. P. Low, "Dynamic Message Ferry Route (dMFR) for Partitioned MANETs," in *Communications and Mobile Computing (CMC), 2010 International Conference on*, vol. 3, april 2010, pp. 447–451.

[WLRS05] S. Y. Wong, J. G. Lim, S. Rao, and W. Seah, "Density-aware hop-count localization (DHL) in wireless sensor networks with variable density," in *Wireless Communications and Networking Conference, 2005 IEEE*, vol. 3, march 2005, pp. 1848–1853 Vol. 3.

[yam12] "YAML," Website, April 2012, available online: http://www.yaml.org.

[YKC04] P. Yao, E. Krohne, and T. Camp, "Performance comparison of geocast routing protocols for a MANET," in *Computer Communications and Networks, 2004. ICCCN 2004. Proceedings. 13th International Conference on*, 11-13 2004, pp. 213–220.

[YNK02] S. Yi, P. Naldurg, and R. Kravets, "A Security-Aware Routing Protocol for Wireless Ad Hoc Networks," in *ACM SYMPOSIUM ON MOBILE AD HOC NETWORKING & COMPUTING (MOBIHOC)*, 2002, pp. 286–292. [Online]. Available: http://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.6.9961

[zer12] "ZeroMQ," Website, April 2012, available online: http://www.zeromq.org.

List of Figures

1.1 Network partitioning recovery after a base station failure 2

2.1 Wireless networks by communication area, according to [LC05] 6
2.2 Hidden- and exposed-terminal problems 9
2.3 MANET routing algorithms 13

3.1 Disaster management cycle concept (adopted from [TI10]) 20

4.1 Recovery and usage communication planes 36
4.2 System architecture and system borders 38
4.3 Place of the partitioning recovery system in Internet protocol suite . . . 40
4.4 Unit graph model 41

5.1 Example of a simple Click router graph 45
5.2 Example of a simple Click router script 45
5.3 Connection states from communication partner tracking table 51
5.4 Problem detector within the AODV Click router graph 52
5.5 Reference points for the linearization 58
5.6 Linearization error by the distance calculation for different latitudes . . 59
5.7 Coordinate option within an IPv4 header 61
5.8 Coordinate option within an IPv6 hop-by-hop extension header 62
5.9 Extension of IPv4 headers in Click 64
5.10 Two network partitions reconnected by an additional node 65
5.11 Geography-aware routing protocols, based on [BNSM09] 66
5.12 Local maxima problem by the geographic routing 71
5.13 Types of the node's neighbors for the Heuristic Search algorithm 77
5.14 Example of a complex network topology with a gap 77
5.15 Average packet count (overhead) (a) and average searching time (b) for NNS procedure with different base algorithms 80
5.16 Click router graph for the NNS implementation 81
5.17 Example of the simple YAML configuration file for SCL 84
5.18 Data Access Module (DAM) 85
5.19 A Voronoi diagram of 11 points in the Euclidean plane (from [AK00]) . 87
5.20 Extended Voronoi diagram with specified node placement positions . . 89
5.21 Possible last known positions of the communication partner node . . . 90
5.22 Comparison of the node placement strategies with and without available global knowledge 91

5.23 Mission request (`MREQ`) and mission reply (`MREP`) messages format . . . 92
5.24 Three levels of aggregation 95

6.1 NS-2: Basic simulation model 99
6.2 NS-3: Basic simulation model, taken from [Hen10a] 100
6.3 OMNeT++: Basic simulation model 103
6.4 Mobile host model in OMNeT++/INETMANET 104
6.5 NetViewer GUI 109
6.6 Typical scenario for the testbed 110
6.7 Demonstrator network architecture 111
6.8 Scenario for the node placement demonstrator 112
6.9 Multicopter 114
6.10 Demonstrator headquarter server architecture 115

List of Tables

2.1 Proactive routing protocols examples 14
2.2 Reactive routing protocols examples 14
2.3 Hybrid routing protocols examples 15

3.1 Comparison of the network partitioning recovery systems. *Part I* . . . 28
3.2 Comparison of the network partitioning recovery systems. *Part II* . . . 29

5.1 List of available AODV Click elements 46
5.2 Performance comparison of LKM and Click implementations for IPv4 header extension 63
5.3 Algorithms for the calculation of Voronoi diagrams 87

6.1 Comparison of NS-2, NS-3 and OMNeT++ 105

List of Abbreviations

ACK ..... **Ack**nowledgement
AoA ..... **A**ngle-**o**f-**A**rrival
AODV ..... **A**d-hoc **O**n-demand **D**istance **V**ector Routing
AP ..... **A**ccess **P**oint
API ..... **A**pplication **P**rogramming **I**nterface
AWDS ..... **A**d-hoc **W**ireless **D**istribution **S**ervice
BATMAN ..... **B**etter **A**pproach **T**o **M**obile **A**d-hoc **N**etworking
CBR ..... **C**onstant **B**it **R**ate
CEDAR ..... **C**ore **E**xtraction **D**istributed **A**d hoc **R**outing
CGSR ..... **C**lusterhead **G**ateway **S**witch **R**outing
CI ..... **C**onnectivity **I**ndex
CLDP ..... **C**ross-**L**ink **D**etection **P**rotocol
CORBA ..... **C**ommon **O**bject **R**equest **B**roker **A**rchitecture
CTS ..... **C**lear **T**o **S**end
DAG ..... **D**estination-oriented directed **A**cyclic **G**raph
DAM ..... **D**ata **A**ccess **M**odule
DDFS ..... **D**istributed **D**epth-**F**irst **S**earch
DMCC ..... **D**etection algorithm based on **M**idpoint **C**overage **C**ircle
DREAM ..... **D**istance **R**outing **E**ffect **A**lgorithm
DSR ..... **D**ynamic **S**ource **R**outing
DTN ..... **D**elay **T**olerant **N**etwork
DYMO ..... **Dy**namic **M**ANET **O**n-demand Routing
EARBALE ..... **E**nvironment-**A**ware **R**SS-**Ba**sed **L**ocation **E**stimation
EPPD ..... **E**ventually **P**erfect **P**artition **D**etector
FORP ..... **F**low-**O**riented **R**outing **P**rotocol

FTP ..... **F**ile **T**ransfer **P**rotocol

GAMER ..... **G**eocast **A**daptive **M**esh **E**nvironment for **R**outing

GCR ..... **G**eo**c**ast **R**egion

GEDIR ..... **Ge**ographic **Di**stance **R**outing

GeoPPRA ..... **Geo**graphic-aware **P**rioritized **P**heromone-aided **R**outing **A**lgorithm

GFG ..... **G**eographic-**F**orwarding-**G**eocast routing

GFG ..... **G**reedy-**F**ace-**G**reedy Routing Protocol

GFPG ..... **G**eographic-**F**orwarding-**P**erimeter-**G**eocast routing

GG ..... **G**abriel **G**raph

GGP ..... **G**eometry-driven **G**eocasting **P**rotocol

GLONASS ..... **Glo**bal **Na**vigation **S**atellite **S**ystem

GPS ..... **G**lobal **P**ositioning **S**ystem

GPSR ..... **G**reedy **P**erimeter **S**tateless **R**outing

GS Mobicom ..... International **G**raduate **S**chool on **Mobi**le **Com**munications

HBFD ..... **H**eart**b**eat **F**ailure **D**etector

HRPLS ..... **H**ybrid **R**outing **P**rotocol for **L**arge **S**cale MANETs with Mobile Backbones

HS ..... **H**euristic **S**earch

ICMP ..... **I**nternet **C**ontrol **M**essage **P**rotocol

LAN ..... **L**ocal **A**rea **N**etwork

LAR ..... **L**ocation-**A**ided **R**outing

LBM ..... **L**ocation-**B**ased **M**ulticast algorithm

MAC ..... **M**edium **A**ccess **C**ontrol

MAN ..... **M**etropolitan **A**rea **N**etwork

MANET ..... **M**obile **A**d-hoc **Net**work

MREQ ..... **M**ission **Req**uest

NNS ..... **N**earest **N**odes **S**earch

NS-x ..... **N**etwork **S**imulator **2** or **3**

OFGP ..... **O**bstacle-**F**ree **G**eocasting **P**rotocol

OLSR ..... **O**ptimized **L**ink-**S**tate **R**outing

OTcl ..... **O**bject **T**ool **C**ommand **L**anguage

PADRA ..... **Pa**rtitioning **D**etection and **R**ecovery **A**lgorithm

PAN ..... **P**ersonal **A**rea **N**etwork

PDS ..... **P**artitioning **D**etection **S**ystem

PGM ..... **P**ragmatic **G**eneral **M**ulticast

PPE ..... **P**ing-**P**ong **E**ffect

PSR ..... **P**ower-aware **S**ource **R**outing

PUMA ..... **P**rotocol for **U**nified **M**ulticasting through **A**nnouncements

QoS ..... **Q**uality-**o**f-**S**ervice

RNG ..... **R**elative **N**eighborhood **G**raph

RPC ..... **R**emote **P**rocedure **C**all

RREQ ..... **R**oute **Req**uest

RSS ..... **R**eceived **S**ignal **S**trength

RTS ..... **R**equest **T**o **S**end

SAR ..... **S**ecurity-**a**ware **R**outing

SCL ..... **S**ignaling and **C**ommunication **L**ink

SENCAST ..... **S**calable Protocol for Large Ad-hoc **E**mergency **N**etwork for Uni**cast**ing and Multi**cast**ing

SQL ..... **S**tructured **Q**uery **L**anguage

SSR ..... **S**calable **S**ource **R**outing

TCP ..... **T**ransmission **C**ontrol **P**rotocol

TDoA ..... **T**ime-**D**ifference-**o**f-**A**rrival

ToA ..... **T**ime-**o**f-**A**rrival

TORA ..... **T**emporally-**O**rdered **R**outing **A**lgorithm

TTL ..... **T**ime **T**o **L**ive

UAV ..... **U**nmanned **A**erial **V**ehicle

UDP ..... **U**ser **D**atagram **P**rotocol

USB ..... **U**niversal **S**erial **B**us

VBDD ..... **V**ector-**B**ased **D**isconnection **D**etector

WAN ..... **W**ide **A**rea **N**etwork

WDN ..... **W**ide area **D**isaster information **N**etwork

WGS 84 ..... **W**orld **G**eodetic **S**ystem **1984**
WiMAX ..... **W**orldwide **I**nteroperability for **M**icrowave **A**ccess
WLAN ..... **W**ireless **L**ocal **A**rea **N**etwork
WRP ..... **W**ireless **R**ecovery **P**rotocol
WSN ..... **W**ireless **S**ensor **N**etwork
YAML ..... **Y**AML **A**in't **M**arkup **L**anguage
ZHLS ..... **Z**one-based **H**ierarchical **L**ink **S**tate routing protocol
ZRP ..... **Z**one **R**outing **P**rotocol

Theses

1. Mobile Ad-hoc Networks (MANETs) play a vital role in disaster recovery measures since they provide the ability of quick information exchange, which can be necessary for coordination of rescue teams or of members of a rescue team. Sometimes the humans' lives can depend on the properly working communication network.

2. One of the important mechanisms of MANETs is multi-hop routing. Due to the routing, large areas can be covered by a small number of network nodes.

3. The most famous problem of MANETs is network partitioning. Due to nodes' movement, the network can be split into several isolated groups of nodes, which are unable to communicate with each other. The best solution of this problem is the placement of additional nodes. In opposite to other solutions, it does not divert the network resources, which are used for disaster recovery, in order to reconnect the network partitions.

4. Node placement positions can be estimated using Nearest Nodes Search approach in a self-organizing manner using only the knowledge available locally for the nodes.

5. An optimal placement position calculation can be achieved if additional information about the network topology was collected in advance.

6. The proposed system is able to work with networks of mobile devices of any type, since it does not require any hardware from the network being repaired and is a pure software solution potentially working with any underlying technology supporting wireless multi-hop communication.

7. As an additional node, any device can be used which supports the used network communication technology and which is able to change its own position on the request of the system.

8. The solution ideas proposed in this work are used within a joint network recovery demonstrator of the International Graduate School on Mobile Communication (GS Mobicom)

Monday 10th February, 2014, Ilmenau M.Sc. Mikhail Tarasov